Clustered Firefly Algorithm for Global Parameter Estimation in Cyber-Physical Systems

Applications in Wastewater Treatment and Residential Heating

Master Thesis

Hannes Gebauer

Aalborg University
Computer Science

AALBORG UNIVERSITY
STUDENT REPORT

Computer Science
Aalborg University
https://www.aau.dk/

Title:
Clustered Firefly Algorithm for Global Parameter Estimation in Cyber-Physical Systems: Applications in Wastewater Treatment and Residential Heating

Theme:
Computer Science, Parameter Estimation

Project Period:
Spring Semester 2025

Project Group:
cs-25-sv-10-05

Participant(s):
Hannes Gebauer

Supervisor(s):
Kim Guldstrand Larsen

Page Numbers: 83

Date of Completion:
June 11, 2025

Abstract:

This thesis presents a novel variant of the Firefly Algorithm—called the *Clustered Firefly Algorithm* (cFA)—for efficient parameter estimation in dynamic models used within the context of Model Predictive Control (MPC). By integrating k-Means clustering with the original Firefly Algorithm, the proposed method reduces time complexity while maintaining strong global search capabilities.
Benchmark experiments across several standard optimization functions demonstrate that the cFA converges faster and more reliably to high-quality solutions compared to the original Firefly Algorithm and other baseline methods.
The algorithm's effectiveness is further validated through two real-world applications: parameter estimation for a theoretical wastewater treatment plant (ASM1 model) and a thermal building model used for heat pump control. Results indicate that the algorithm handles complex, multi-modal search landscapes effectively. While the results are promising, further experimental validation is recommended to fully assess the cFA's applicability across diverse MPC scenarios.

Summary

The energy sector is one of the largest contributors to climate change. To mitigate its impact, more efficient energy solutions must be developed. One strategy to improve energy efficiency in existing technologies is the use of advanced real-time computational control methods that derive optimal, energy-efficient control actions.

A prominent framework for implementing such strategies is *Model Predictive Control* (MPC). MPC relies on a dynamic model of the system to predict the future trajectory of system states. Based on these predictions, an optimization algorithm computes control actions that are applied to the real system. However, the effectiveness of MPC strongly depends on the accuracy of the underlying model, as control decisions are derived from its predictions.

Many such models are parameterized, and their performance depends on the accurate identification of model parameters. Therefore, accurate parameter estimation is essential for achieving effective and reliable control.

This thesis presents a novel variant of the Firefly Algorithm—called the *Clustered Firefly Algorithm* (cFA)—for efficient and accurate parameter estimation in the context of Model Predictive Control. The proposed method integrates the original Firefly Algorithm with k-means clustering to reduce time complexity with respect to objective function evaluations, which are typically a computational bottleneck in simulation-based optimization problems. At the same time, the algorithm retains strong global search capabilities. Several additional modifications to the original FA are introduced to improve convergence behavior and reduce runtime.

It is formally shown that the cFA reduces the per-generation time complexity from $O(n^2)$ to $O(n^{3/2})$ compared to the original Firefly Algorithm.

Furthermore, benchmark experiments demonstrate that the cFA outperforms both the original FA and a Grid Search baseline in terms of solution quality and convergence speed across a range of standard test functions.

To evaluate the algorithm in practice, the cFA is applied to two real-world case studies.

The first case study focuses on estimating kinetic parameters in a theoretical wastewater treatment plant based on the Activated Sludge Model No. 1 (ASM1). This work is conducted in collaboration with *Dansk Hydraulisk Institut* (DHI). The results demonstrate the cFA's ability to navigate complex, non-linear optimization landscapes. While promising, these results are based on synthetic data, as no real measurements were available. To evaluate the algorithm's practical applicability, more realistic experiments would be necessary.

The second case study addresses thermal parameter estimation in residential buildings for heat pump control, in collaboration with the company *CEDAR*. Real-world data from a building in northern Denmark is used to estimate thermal parameters for multiple rooms in an interconnected house model. For each room, a separate set of parameters is estimated, capturing individual thermal dynamics. The results show that the cFA can successfully estimate parameters for simplified models, even under noisy data conditions.

Although parameters were estimated successfully for most rooms, further experiments revealed that some parameter sets did not generalize well, suggesting overfitting. Additional experimentation with a more refined setup is needed to fully assess the algorithm's applicability for thermal parameter estimation in MPC contexts.

Future work could include integrating the cFA into a complete MPC framework to investigate the practical impact of global parameter estimation on control objectives, such as reducing energy costs.

Preface

This thesis was written as part of the Master's programme in Computer Science (IT) at Aalborg University during the spring semester of 2025. The work focuses on efficient parameter estimation for control-oriented modeling in cyber-physical systems, with applications in wastewater treatment and residential heating.

The wastewater treatment case study was conducted in collaboration with *Dansk Hydraulisk Institut* (DHI), and the residential heating case study was conducted in collaboration with *CEDAR*, who provided real-world data from a building in northern Denmark.

I would like to thank my thesis supervisor, Kim Guldstrand Larsen, for his valuable guidance and support throughout the project. I also wish to thank Christopher Eugen Gaszynski, Enrico Ulisse Remigi, and Trine Dalkvist from DHI for their helpful input and continued cooperation. Additionally, I would like to thank Peter Gjøl Jensen from CEDAR for providing the building data and for his assistance throughout the heat pump case study.

Aalborg University June 11, 2025

H. Gebauer

Hannes Gebauer

<hgebau23@student.aau.dk>

Contents

Chapter 1

Introduction

Climate change is widely recognized as one of the most pressing challenges of the twenty-first century. Its impacts are already visible today, manifesting among other things through biodiversity loss, more frequent and intense extreme weather events, and increasing food insecurities [1]. If global warming continues unabated, these risks and adverse effects are expected to intensify.

In 2015, the international community adopted the *Paris Agreement* under the United Nations Framework Convention on Climate Change, committing to limit global temperature rise to well below 2°C above pre-industrial levels, and to pursue efforts to restrict it to 1.5°C [2].

However, in a recent report the Intergovernmental Panel on Climate Change (IPCC) warns that current climate policies are inadequate to meet these targets [1]. Without immediate and substantial action, global warming is likely to exceed 1.5°C within the twenty-first century. The report further notes that while limiting warming to 2°C remains more plausible, even this target is not assured under existing commitments. Exceeding these temperature limits would significantly heighten the risk of irreversible biodiversity loss and long-term harm to human health, livelihoods, and infrastructure. With this in mind, the urgency of identifying immediate and effective mitigation strategies for climate change is evident.

The energy sector is one of the largest contributors to climate change, accounting for approximately 73% of global greenhouse gas emissions [3]. This makes it a primary target for strategies aimed at mitigating climate change.

As the energy demand from industry and other sectors is not expected to decline in the foreseeable future, alternative approaches are required to reduce emissions. One widely discussed solution is the transition to carbon-neutral renewable energy sources. However, building a sufficient infrastructure for large-scale renewable energy deployment requires significant time and investment.

A complementary and more immediately actionable strategy is to improve the efficiency of existing energy systems. For instance, the International Energy Agency (IEA) highlights that significantly enhancing energy efficiency is essential for achieving carbon neutrality by 2050 [4].

One effective way to improve energy efficiency in existing systems is through the implementation of advanced control strategies. A particularly promising framework for this purpose is *Model Predictive Control* (MPC). MPC relies on a dynamic model of the system to compute control actions that optimize a given objective, such as minimizing energy consumption.

Unlike static control strategies, MPC continuously updates its control inputs by solving an optimization problem for each time period, using the most recent system information [5]. This receding horizon approach allows MPC to adapt to changes in the system and environment, making it more robust and responsive than fixed policy strategies.

Depending on the system dynamics and application requirements, MPC can be implemented using different types of models and optimization strategies.

Generally, models can be categorized into three types [6]:

- *White-box models* are derived from first principles and are based on physical laws such as conservation of mass, energy, and momentum. They are typically very accurate and interpretable, making them suitable for well-understood systems. However, they require detailed system knowledge and significant effort to model. Additionally, simulation times can be long, which may pose challenges for deriving real-time advanced control strategies, for example using MPC.

- *Black-box models* rely purely on data to learn the mapping from inputs to outputs. Common examples include neural networks and other machine learning algorithms. These models require minimal domain knowledge and can model highly complex systems, but they demand large amounts of high-quality data. They also lack interpretability, making it difficult to verify whether the model behaves realistically across all scenarios. This can pose risks, particularly in safety-critical applications.

- *Grey-box models* combine elements of both white-box and black-box approaches. They incorporate simplified physical models (e.g., ordinary differential equations) while using data to identify unknown parameters. Although typically less accurate than full white-box models, they offer significantly faster simulation times and retain a degree of physical interpretability. This makes them a practical compromise for many real-world applications, especially in control and optimization tasks.

These different model types can be combined with a variety of optimization techniques to compute near-optimal control strategies for the underlying system. In the following, some examples are presented.

For instance, Linear Programming (LP) and Quadratic Programming (QP) can be used to minimize linear or quadratic cost functions, respectively, subject to linear constraints and system dynamics [5]. A limitation of this approach is that the underlying system model must be linear or appropriately linearized, which restricts its applicability with more complex or highly nonlinear systems.

Another approach to deriving control strategies is through Reinforcement Learning (RL), which learns by interacting with the environment or model. A notable example applied in MPC contexts is UPPAAL Stratego [7]. In this approach, the system is typically modeled as a grey-box using hybrid timed automata, and Stratego synthesizes a near-optimal control strategy based on a given objective using a partition-refinement-based Q-learning algorithm [8]. A key advantage of this method is that it can be applied to complex, nonlinear systems without requiring linearization or differentiability of the model.

In contrast to MPC, many neural network-based controllers aim to learn a fixed optimal policy during training. While effective in static environments, they often require retraining to adapt to changes in system dynamics or operating conditions. This limitation highlights the advantage of adaptive, model-based approaches like MPC, which continuously optimize control actions in response to updated system information.

A prerequisite for MPC is a sufficiently accurate model of the system, as control decisions are directly informed by the model's predictions.

In the context of this work, the focus is placed on grey-box models. While these models offer a structured framework for capturing system dynamics, their accuracy depends on identifying the correct parameter values. Without proper parameter estimation, even a well-designed model may fail to reflect the real behavior of the system. This underlines the importance of parameter estimation methods for achieving effective control.

To illustrate the concept of MPC, Figure 1.1 presents an overview of the MPC framework in the context of residential building heating. The figure is inspired by the study conducted by *Hasrat et al.* [9], where significant energy savings were achieved by combining a grey-box building model with regularly updated parameter estimation and reinforcement learning using UPPAAL Stratego.

The figure highlights the two main components of the MPC framework: the main control loop and the parameter estimation loop.

In the parameter estimation loop, historical measurement data from the building are used to identify or update the parameters of a thermal house model.

In the main MPC loop, real-time measurements—together with predictive inputs such as weather forecasts and day-ahead energy prices—are used as inputs to the model to simulate future system behavior. An optimization algorithm then computes a control strategy that minimizes a predefined objective, such as energy use or operational cost. This strategy is forwarded to the building's controllers, which implement the next control action.

The main loop is repeated for each control period, with the control strategy typically recalculated after each action, enabling it to dynamically adapt to changing conditions.

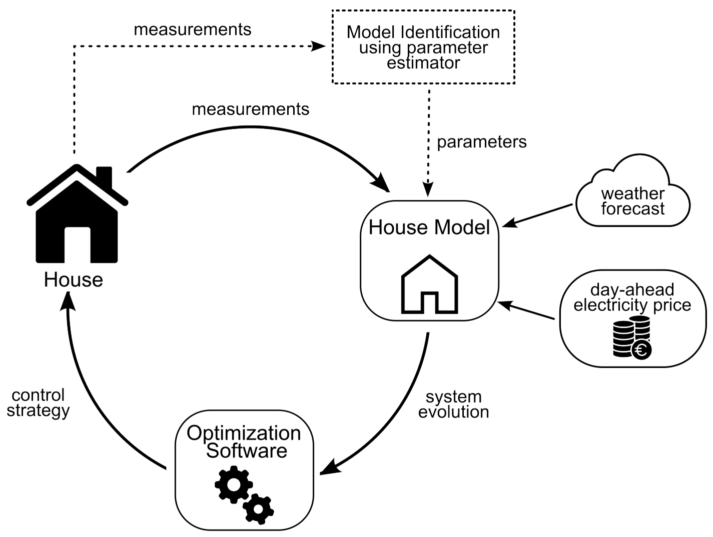


Figure 1.1: Model Predictive Control framework for heating cost reduction in residential buildings.

In contrast, the parameter estimation loop is executed less frequently, since the underlying physical parameters (e.g., thermal properties) typically change more slowly over time. This dual-loop structure enables MPC to provide a continuously updated, near-optimal control strategy that responds both to environmental changes and to system uncertainties.

The core focus of this thesis lies in the parameter estimation loop—specifically, in the development of an algorithm for identifying model parameters. To this end, the thesis introduces a new variant of a nature-inspired global optimization algorithm—the Firefly Algorithm (FA) [10]–for efficient and qualitative parameter estimation. The FA was first proposed by *Yang* in 2008 and is inspired by the bioluminescent flashing behavior observed in fireflies. In the algorithm, each firefly represents a candidate solution in the optimization space. Fireflies are attracted to others with higher "brightness," which corresponds to better objective function values. This attraction mechanism guides the swarm through the search space by balancing random exploratory movements with directed exploitation of promising regions. The algorithm has been shown to outperform other nature-inspired global optimizations algorithms in terms of efficiency and success rate–meaning how often the global maximum was found on a range of benchmark problems [10].

Although the FA is not among the most recent nature inspired metaheuristics, it remains widely used due to its simplicity, flexibility, and competitive performance across a broad range of application domains. In a recent survey on the FA [11], *Li et al.* highlight its use in diverse fields, including general optimization, classification, industrial process control, and robotics, among others. They emphasize that the algorithm's simplicity makes it particularly amenable to domain-specific modifications. This adaptability makes FA a strong choice for this thesis, as it can be customized to fit within the MPC framework.

To this end, the thesis proposes a modified variant called the *Clustered Firefly Algorithm* (cFA). This approach integrates the original FA with a clustering technique to improve computational efficiency while maintaining strong exploratory performance.

The improved time complexity of the cFA is formally analyzed, and its performance is benchmarked on a range of optimization problems. The results demonstrate that cFA generally converges more rapidly and consistently to better solutions compared to both the original FA and a grid-search baseline.

Additionally, the cFA is applied to estimate parameters in two real-world use cases that significantly contribute to global energy consumption. These use cases have been studied in collaboration with two companies.

The first case involves wastewater treatment plants (WWTP) and was conducted in collaboration with **Dansk Hydraulisk Institut** (DHI)[1]. WWTPs are estimated to account for approximately 1% of global energy usage [12]. In this context, the cFA is used to estimate sensitive parameters in a theoretical plant modeled using the Activated Sludge Model No. 1 (ASM1) [13]. Another student group is working on the integration of MPC in WWTPs with the aim of reducing energy costs and improving effluent quality. The parameter estimation approach developed in this thesis is complementary to that effort and could be integrated into their MPC framework to identify accurate model parameters.

The second case focuses on residential buildings heated by heat pump systems. Residential energy consumption is estimated to contribute around 10.9% to global energy use [3], making it a promising target for efficiency improvements. The cFA is applied to estimate parameters for a grey-box thermal model of a residential building located in northern Denmark, using real-world data provided by the company **CEDAR**[2].

For both use cases, the generalization performance of the estimated parameters is evaluated on unseen data to assess the robustness of the identified models—an essential requirement for reliable performance in an MPC framework.

Overall, the cFA demonstrates promising results in both use cases as an effective method for parameter identification. However, further improvements in experimental design may be necessary to fully assess its practical applicability in real-world control scenarios.

[1]DHI `https://www.dhigroup.com/`

[2]CEDAR: `https://cedar-heat.dk/`

Chapter 2

Related Methods and Related Work

Parameter estimation in complex systems has been extensively studied across scientific and engineering disciplines. A wide range of optimization techniques have been developed to identify model parameters that best reproduce observed data, from classical statistical methods to modern metaheuristics and machine learning approaches.

This section provides an overview of selected methods and relevant literature, with a particular focus on the two application domains addressed in this thesis: kinetic parameter estimation in wastewater treatment plants (WWTPs) and thermal parameter estimation in residential building systems.

2.1 Related Methods

Gradient-Based Approaches Gradient-based optimization methods are widely regarded as foundational and effective techniques for parameter estimation in dynamical systems, particularly when derivatives of the objective function are available or can be approximated. These methods iteratively adjust model parameters to minimize or maximize an objective function by following the direction of the gradient—corresponding to the steepest descent in minimization problems. While highly efficient for smooth and well-behaved problems, gradient-based methods guarantee convergence to a global optimum only in the case of convex objective functions. For non-convex problems, they may converge to local optima, as gradient information alone does not convey global structure [14].

When analytical gradients are unavailable, they can be approximated numerically using finite differences. However, for complex systems, this approach can be computationally expensive and sensitive to both truncation and perturbation errors [15]. These limitations have motivated the use of alternative optimization and parameter estimation techniques that do not rely on gradient information.

Bayesian Inference Bayesian inference methods estimate a full probability distribution over the model parameters rather than a single point estimate, as is typically obtained with gradient-based approaches [16]. This probabilistic perspective allows for explicit quantification of uncertainty in the parameter estimates, making Bayesian methods particularly valuable for stochastic systems that include randomness or measurement noise.

At the core of Bayesian inference is the computation of the posterior probability distribution $P(\theta \mid y)$, which represents the probability of the parameters θ given the observed data y. Bayes' Theorem defines this relationship as:

$$P(\theta \mid y) = \frac{P(y \mid \theta)P(\theta)}{P(y)}$$

where $P(y \mid \theta)$ is the likelihood, that is, the probability of the observation y given the parameters θ; $P(\theta)$ is the prior; and $P(y)$ is the marginal likelihood or evidence.

For complex models, computing $P(y)$ analytically is often intractable, as it requires integration over all possible parameter values, so approximate methods such as Markov Chain Monte Carlo (MCMC) sampling [16] or variational inference [17] are used to estimate the posterior. While Bayesian inference provides a more comprehensive view of parameter uncertainty, it is computationally demanding—particularly in simulation-based models—due to the large number of samples required for convergence.

Machine Learning-Based Approaches With the advancement of data-driven modeling, several machine learning-based approaches for parameter estimation have emerged. At their core, these methods learn mappings from inputs to outputs by optimizing model parameters using available data. Examples range from classical models such as linear regression and support vector machines to more complex architectures like deep neural networks [18].

Neural networks, particularly deep neural networks, have received significant attention in recent years due to their ability to approximate arbitrary nonlinear functions given sufficient complexity (i.e., number of layers and neurons) [19]. This universal function approximation property makes them highly expressive, but also susceptible to overfitting, especially in low-data regimes. Moreover, their high parameter complexity often results in limited interpretability, which is why they are commonly referred to as "black-box models." Ensuring good generalization typically requires large amounts of training data, which can be a limiting factor in scientific applications where data is scarce or expensive to obtain.

To address these limitations, the framework of *Physics-Informed Neural Networks* (PINNs) has been proposed [20]. PINNs incorporate physical knowledge—often in the form of partial differential equations—into the training process by adding physics-based terms to the loss function. This regularization by prior knowledge enables better generalization even with less data, and allows the model to respect known physical constraints.

Global Optimization Algorithms Another class of widely used optimization algorithms for parameter estimation is the class of global optimization methods. In contrast to gradient-based techniques and other local optimizers, global methods aim to explore the entire parameter space to identify the globally optimal solution, rather than converging to the nearest local minimum. In the context of parameter estimation, this means searching for the parameter configuration that minimizes an objective function—such as the error between simulated model output and observed data.

Since an exhaustive search is computationally infeasible for all but the lowest-dimensional problems, global optimization relies on heuristic search strategies to efficiently explore the parameter space. A notable advantage of most global optimization algorithms is their ability to operate without gradient information, making them particularly suitable for problems where the objective function is non-differentiable, noisy, or expensive to evaluate.

Many of these algorithms draw inspiration from natural phenomena. For example, the *Genetic Algorithm* (GA) mimics the process of natural selection by evolving a population of candidate solutions over generations [21]. Each individual is assigned a fitness score based on the objective function, and new candidates are generated using selection, crossover, and mutation operations. Another prominent example is *Particle Swarm Optimization* (PSO) [22]. This algorithm is inspired by swarming behavior observed in nature, such as bird flocking or fish schooling. Similar to genetic algorithms, PSO maintains a population of candidate solutions—called particles—that evolve over several iterations. Each particle adjusts its position in the search space based on its own best-known position and the best-known position found by the entire swarm. This combination of individual knowledge and *social sharing* allows the swarm to efficiently explore the search space and converge toward high-quality solutions.

The parameter estimation algorithm developed in this thesis is based on the *Firefly Algorithm* [10] (FA), which is a swarm based optimization algorithm similar to PSO. As already stated in the introduction, it has been shown to outperform other nature inspired global optimization algorithms such as GA and PSO in terms of efficiency and solution quality [10]. The FA will be described in detail in Section 4.1.

2.2 Related Work

The parameter estimation methods discussed above have been applied in a wide range of scientific and engineering domains. This thesis focuses on two specific application areas: kinetic parameter estimation in WWTPs and thermal parameter estimation in residential building systems. In the following, we review selected contributions from the literature related to these two domains, with an emphasis on how the aforementioned methods (especially the global optimization methods) have been applied in practice.

A recent review by *Deepak et al.* [23] discusses the application of various nature-inspired global optimization algorithms to a broad range of problems in wastewater treatment, including process design, control, and parameter estimation. Their findings indicate that these algorithms can offer significant improvements in performance, robustness, and modeling flexibility.

One important area of application is control optimization. *Selamat et al.* [24] and *Choo et al.* [25] employed PSO to automatically tune PID controllers in activated sludge processes, focusing on control variables such as dissolved oxygen and nitrate concentrations. Both studies reported improved dynamic response and reduced tuning effort compared to traditional heuristic tuning methods.

Another relevant domain is the integrated optimization of both process control and plant design. For example, *Schlüter et al.* [26] proposed an *Extended Ant Colony Optimization Algorithm*—a global optimization method based on swarm intelligence—for simultaneous optimization of control setpoints and structural configurations in WWTPs. Their approach optimized both continuous parameters (e.g., aeration rates, controller gains) and discrete design choices (e.g., number of tanks, placement of anoxic zones). The results showed superior performance and cost-efficiency across multiple benchmark scenarios compared to conventional strategies.

In addition to control and design, nature-inspired algorithms have been successfully applied to parameter estimation tasks, which are a central focus of this thesis. For instance, *Khoja et al.* [27] and *Du et al.* [28] applied variants of the *Cuckoo Search Algorithm*–an algorithm close to the principles of GA–to estimate kinetic and stoichiometric parameters in activated sludge models. These studies demonstrated that such metaheuristic approaches can achieve accurate parameter identification and robust model fitting, even in highly nonlinear and multimodal optimization landscapes.

Overall, the literature demonstrates that nature-inspired global optimization algorithms are effective tools across a range of wastewater treatment applications. However, a common drawback of these methods is their relatively high computational cost due to the need for extensive exploration of the parameter space. This thesis aims to address this limitation by developing a novel variant of the well-established Firefly Algorithm—the *Clustered Firefly Algorithm*—which achieves lower time complexity while maintaining strong parameter estimation performance.

While the wastewater treatment domain offers a well-established setting for exploring parameter estimation methods, similar challenges arise in the context of residential building systems—particularly in modeling and optimizing thermal behavior. Accurate estimation of thermal parameters is crucial for developing energy-efficient control strategies. In the following, we review relevant literature addressing parameter estimation in residential building systems, with a focus on data-driven modeling and nature-inspired optimization techniques.

In one study, *Saryazdi et al.*[29] developed a data-driven optimization framework for residential buildings in hot climates, using an artificial neural network (ANN) as a surrogate model to approximate building performance simulations. The ANN was trained to predict cooling energy consumption, discomfort hours, and carbon emissions based on

various building design and operational parameters. They then applied a multi-objective genetic algorithm to identify optimal parameter combinations, demonstrating significant reductions in all three objectives. While effective they do not use parameter estimation in the context of more complex control strategies.
In a another study, *Rivera et al.* [30] employ *RC networks* as grey-box models to represent the thermal dynamics of a residential building located in a tropical climate. They define several candidate RC network topologies based on physical insight and use MATLAB's `greyest` function—an algorithm that selects the most suitable among several local optimization methods—to estimate model parameters. Due to the local nature of the optimizer, the authors run the parameter identification multiple times with varying initial guesses to increase the likelihood of converging to a global minimum. The final identified model is then used to automatically tune a PID controller, which successfully regulates indoor temperature to a predefined setpoint. Although the study shows promising results, their model assumes the house to be a single thermal zone, which compromises its applicability for fine-grained or more advanced control strategies.
Hasrat et al. [9] developed a thermal grey-box model of a multi-zone residential house equipped with a heat pump system. In their approach, each room is modeled with its own set of parameters and thermal interactions with neighboring rooms are considered. Each room's thermal dynamics are represented using three state variables: room air temperature, floor temperature, and envelope temperature. Parameter estimation is performed using *CTSM-R* [31], a gradient-based local optimization tool for continuous-time stochastic systems. The identified models are then incorporated into a broader MPC framework, where parameters are periodically re-estimated based on recent data. The identified models are used with *Uppaal Stratego* [7], a tool that integrates stochastic hybrid systems and reinforcement learning to synthesize optimal control strategies. Their case study reports energy cost savings of up to 46–49% compared to a baseline control strategy.

The room-level temperature model developed by *Hasrat et al.* is also employed in this thesis as the basis for thermal parameter estimation.

Among the reviewed studies, *Saryazdi et al.* focused on data-driven optimization for building design, using a global nature-inspired algorithm—a genetic algorithm—to identify optimal design parameters. In contrast, both *Rivera et al.* and *Hasrat et al.* addressed control applications using grey-box models in combination with local optimization methods for parameter estimation.

In this thesis, parameter estimation is pursued in a control-oriented context using a global optimization method—the previously mentioned *Clustered Firefly Algorithm*. This algorithm is applied to identify the parameters of the grey-box thermal model based on real-world data.

Compared to local optimizers, the proposed global metaheuristic optimization algorithm enables broader exploration of the parameter space, increasing the likelihood of avoiding suboptimal local optima and improving the robustness and accuracy of model identification. Although the global nature of the algorithm entails higher computational cost, this trade-off is justified by its enhanced reliability and potential for more precise parameter estimates.

Chapter 3

Problem Statement

Improving energy efficiency is a critical step toward achieving carbon neutrality. As outlined in the introduction, advanced computational control strategies—such as those used to optimize the operation of existing systems—can play a key role in reducing energy consumption. One framework that has gained widespread adoption in such control tasks is *Model Predictive Control* (MPC). However, the effectiveness of MPC depends on the accuracy of the underlying system model. Since many modeling approaches rely on parameterized models, it is essential to employ accurate parameter estimation methods to ensure the model reflects the real system dynamics. As parameter estimation may be performed repeatedly in MPC, the algorithm must also be computationally efficient and robust—meaning it should consistently produce a valid solution and not fail.

In the previous chapter, various optimization methods suitable for parameter estimation tasks were discussed. For the two case studies considered in this thesis—parameter estimation for wastewater treatment plants and for thermal models of residential buildings— nature-inspired global optimization algorithms have shown promise across a range of related problems. However, their global search capabilities come at the cost of increased computational runtime, which can limit their practicality in time-sensitive control settings such as MPC.

One algorithm that has been successfully applied in a wide range of problems and application domains is the *Firefly Algorithm* (FA), which has been shown to outperform other nature-inspired optimization algorithms such as *Genetic Algorithms* (GA) and *Particle Swarm Optimization* (PSO) [10]. While it shares the drawback of increased computational runtime common to global optimization methods, its simplicity allows for straightforward modifications tailored to specific application domains, as mentioned in the introduction. These characteristics make it a promising foundation for addressing the challenge of efficient and accurate parameter estimation in dynamic systems.

This motivates the following problem statement:

> *Can we develop a new variant of the Firefly Algorithm that enables more efficient and accurate global parameter estimation for dynamic system models in the context of Model Predictive Control?*

The following subgoals are defined to address the problem statement:

- Develop a new variant of the FA and formally verify its reduced time complexity.
- Demonstrate that the proposed variant achieves strong optimization performance on a set of benchmark optimization problems.
- Apply the proposed variant to parameter estimation in a theoretical WWTP use case and evaluate its generalization performance on unseen data.
- Apply the proposed variant to parameter estimation in a residential building use case using real-world data; compare its performance to an existing method (CTSM-R) and evaluate its generalization on unseen data.

Chapter 4

Methods

4.1 Firefly Algorithm

The *Firefly Algorithm* (FA) is a nature-inspired metaheuristic search algorithm for multimodal optimization problems, first introduced by *Xin-She Yang* [10]. In the following section, the algorithm is described as outlined by *Yang* in [10].

In nature, fireflies use bioluminescent flashing patterns as a form of communication to attract other fireflies. The light intensity I of the flashing decreases as the distance r from the source increases. As a result, fireflies are only visible to other fireflies within a limited distance.
In the context of the algorithm, a firefly represents a set of parameters to be optimized, while the light intensity corresponds to the "goodness" of these parameters according to an objective function. Fireflies are attracted to other fireflies with higher light intensities, while also undergoing randomized movement to explore the search space. Over time, the fireflies collectively explore the search space through a combination of random motion and attraction toward brighter individuals.
The attractiveness between fireflies is modeled as a function that decreases with distance, reflecting the natural weakening of light intensity over space. This mathematical formulation allows fireflies to be more strongly attracted to nearby brighter fireflies while still retaining some degree of exploration.

Formally, this behavior can be described as follows: Let d be the number of parameters to be optimized. A firefly x is a vector of parameters $x = (x_1, ..., x_d)^T \in \mathbb{R}^d$.
Let f be an objective function $f : \mathbb{R}^d \to \mathbb{R}$. The light intensity of a firefly x is determined by $f(x)$.
To initialize the algorithm, a set of initial fireflies $F = \{x_k | k \in \{1...n\}\}$ is initialized. Over several generations, the fireflies will explore the search space.
In each generation, the light intensity I_i of each firefly $x_i \in F$ is compared to that of every

other firefly $x_j \in F$. If $I_i < I_j$, firefly x_i moves toward x_j based on their mutual attraction, combined with a randomized movement component, updating its own intensity after each movement.
The attractiveness β between two fireflies is defined as a function of the distance r_{ij} between them, starting from an initial attractiveness β_0:

$$\beta(r) = \beta_0 e^{-\gamma r^2}, \tag{4.1}$$

where γ controls the rate at which attractiveness decreases with distance. It should be chosen based on the scale of the optimization parameters and the desired balance between exploration and convergence.

The updated position of a firefly x_i after moving to another firefly x_j is given by:

$$x_i^{new} = x_i + \beta_0 e^{-\gamma r^2}(x_j - x_i) + \alpha \cdot S(rand - \frac{1}{2}), \tag{4.2}$$

where, $\alpha \in [0, 1]$ is the randomization parameter determining the magnitude of random movement, $S \in \mathbb{R}^d$ is a scaling factor that accounts for the different scales of the parameters, and $rand \in [0, 1]^d$ is a uniformly sampled random vector.
It should be noted that the movement of fireflies depends on the order in which they are processed. Even in the absence of randomness, evaluating two identical sets of fireflies in different orders can lead to different outcomes due to the sequential nature of their updates.
Algorithm 1 shows the pseudocode for the original FA.

Algorithm 1 Original Firefly Algorithm (FA)

```
1:  Initialize parameters: $\alpha$, $\gamma$, population size $n$, etc.
2:  Define Objective function: $f(x)$
3:  Initialize $n$ fireflies randomly in the search space
4:  for each generation $i = 1$ to $N_{\text{gen}}$ do
5:      for each firefly $x_i$ do
6:          for each firefly $x_j$ do
7:              if $f(x_j)$ better than $f(x_i)$ then
8:                  Move $x_i$ toward $x_j$ based on attractiveness
9:                  Add random movement controlled by $\alpha$
10:                 Recompute $f(x_i)$
11:             end if
12:         end for
13:     end for
14: end for
15: Return the best firefly found
```

One important consideration when implementing the FA is its computational complexity. In each generation, every firefly compares itself with every other firefly. These comparisons are relatively inexpensive, as the intensity values (i.e., objective function evaluations) are typically precomputed at the start of each generation and after each movement. However, when a firefly moves toward a brighter one, it must perform a new evaluation of the objective function at its updated position. This step is generally the most computationally expensive, particularly when the evaluation involves a simulation, as is the case in this work. In the worst case, each firefly may move toward every other brighter firefly, resulting in up to $O(n^2)$ objective evaluations per generation. Consequently, the overall worst-case time complexity is $O(N_{\text{gen}} \cdot n^2)$, with N_{gen} being the total number of generations. If the objective function is costly to evaluate—as with simulations requiring several seconds per evaluation—this quadratic complexity becomes a significant computational bottleneck, especially for large populations.

This observation motivates several of the modifications introduced in the following sections, which aim to reduce unnecessary objective evaluations and improve convergence efficiency.

The next section, Section 4.2, presents minor modifications to the original algorithm. Following that, Section 4.3 introduces a novel extension of the algorithm based on clustering, designed to further improve its performance.

4.2 Modifications of the Algorithm

In this thesis, several modifications to the original FA are proposed to address the computational and practical challenges specific to this project.
Here, the objective function includes the simulation of the system of interest (either the wastewater treatment plant or the house thermal model). Although individual simulations are relatively fast, the objective function is evaluated many times during each generation, depending on the number of fireflies. As a result, computational efficiency becomes a critical concern.
The modifications to the FA are described in the following.

Population Control

In each generation, the intensity of every firefly is compared with that of all other fireflies. If a firefly encounters another with a better intensity, it moves toward the brighter firefly, and the objective function must be evaluated at its new position.

It is evident that reducing the number of fireflies improves computational efficiency, but at the cost of reduced exploration of the search space. To balance this trade-off, the number of fireflies is gradually reduced over the generations. Fireflies with the lowest intensity are removed to ensure that the most promising individuals continue to explore the search space.

During the optimization, the population size at generation i is updated according to

$$n(i) = \max\left(n_{\min}, \left\lfloor n_{\max}\left(1 - \frac{i}{N_{\text{gen}}}\right)\right\rfloor\right), \tag{4.3}$$

where $n_{\max}$ is the initial number of fireflies, $n_{\min}$ is the minimum allowed number, and N_{gen} is the total number of generations.

α Decay

As shown in Equation 4.2, α controls the magnitude of the random movement of each firefly.
To encourage broad exploration during early generations and more focused convergence in later generations, a decay factor is applied to α at each generation i. The update rule is given by

$$\alpha_{i+1} = \alpha_i \cdot \text{decay\_factor}, \tag{4.4}$$

where decay_factor $\in (0, 1)$ controls the rate of decay.
A higher decay factor (closer to 1) results in slower decay, maintaining randomness longer, while a lower decay factor accelerates convergence.

γ Variation

The parameter γ controls the rate at which the attractiveness between two fireflies decreases with distance, as shown in Equation 4.1.
Varying γ therefore influences how far fireflies are willing to travel toward brighter individuals, balancing exploration and convergence. A low γ increases the effective range of attraction, encouraging broader exploration of the search space. In contrast, a high γ limits attraction to nearby fireflies, promoting local convergence.
The update rule for γ at generation i is given by

$$\gamma_i = \gamma_{\min} + (\gamma_{\max} - \gamma_{\min}) \cdot \left(\frac{i}{N_{\text{gen}}}\right), \tag{4.5}$$

where $\gamma_{\min}$ and $\gamma_{\max}$ are the minimum and maximum values of γ, and N_{gen} is the total number of generations. Increasing γ gradually over the course of the optimization favors global exploration in the early stages and local refinement as convergence progresses.

Distance Measure

In the original formulation of the FA, the distance between two fireflies is computed using the standard *Euclidean distance* in d-dimensional space [10]. However, alternative distance measures may be employed depending on the nature of the optimization problem.

In this project, a *normalized Euclidean distance* is used to account for differing parameter scales. The distance between two fireflies x_i and x_j is defined as

$$r_{ij} = \left\| \frac{x_i - x_j}{S} \right\|, \tag{4.6}$$

where $S \in \mathbb{R}^d$ is a vector containing characteristic scaling factors for each parameter, chosen based on the typical ranges or expected magnitudes of the respective parameters. This normalization ensures that parameters with larger magnitudes do not dominate the distance calculation, thereby promoting balanced exploration of the search space.

Handling of Parameter Bounds

To ensure that the optimization remains within physically meaningful regions, parameter-specific bounds were imposed on the search space. Where available, these bounds were chosen based on expert knowledge or literature values for the respective parameters.

When a firefly's movement would cause it to exceed the allowed bounds, a bounce-back mechanism was employed: instead of clamping the firefly to the boundary, its movement was reflected back into the feasible region. This approach prevents fireflies from accumulating at the search space boundaries, which could otherwise bias the swarm's behavior toward the edges and diminish the algorithm's exploratory capabilities.

Mathematically, the bounce-back correction applied after each movement step (Equation 4.2) is defined for each parameter dimension as

$$x_i = \begin{cases} 2L - x_i & \text{if } x_i < L, \\ 2U - x_i & \text{if } x_i > U, \\ x_i & \text{otherwise,} \end{cases} \tag{4.7}$$

followed by clipping to the feasible interval $[L, U]$ if necessary.

4.3 Clustered Firefly Algorithm

To further improve the performance of the FA, we propose an extension based on clustering, referred to as the *Clustered Firefly Algorithm* (cFA).

This extension organizes the fireflies into groups using K-means clustering and modifies their movement strategy based on cluster-wise information. The goal is to reduce the number of objective function evaluations—since these are typically the most time-consuming—while improving convergence speed by combining local search within clusters with guided migration toward stronger clusters.

The cFA presented in this section is a novel contribution developed as part of this thesis. All algorithmic design elements—including the clustering-based movement strategy, centroid-based global attraction, and dynamic cluster size reduction—were designed and implemented specifically for this work.

Clustering

At the start of each generation, all fireflies are grouped into clusters, which correspond to a subset $C_j \subseteq F$ of the total set of fireflies F. The number of clusters is determined dynamically based on the current number of fireflies and a hyperparameter called *target fireflies per cluster*. This parameter controls the average size of each cluster and affects the number of clusters as follows:

$$n_{\text{cluster}} = \max\left(1, \left\lfloor \frac{n_{\text{fireflies}}}{n_{\text{target}}} \right\rfloor\right) \tag{4.8}$$

where $n_{\text{fireflies}}$ is the current number of fireflies, and n_{target} is the target number of fireflies per cluster.

For each cluster C_j, the centroid c_j is tracked, and an intensity value $I(c_j)$ is assigned based on the mean intensity of the fireflies within the cluster. This is defined as:

$$I(c_j) = \frac{1}{n_c} \sum_{i=1}^{n_c} f(x_i), \quad \text{for all } x_i \in C_j \tag{4.9}$$

where n_c is the number of fireflies in cluster C_j, and $f(x_i)$ is the objective function value (intensity) of firefly x_i.

k-means

In this thesis, *k-means* was chosen as a clustering method. It is a centroid-based algorithm that partitions data into k clusters, minimizing within-cluster variance [32].
The algorithm operates iteratively by first initializing k centroids, typically chosen randomly from the data. Each data point is then assigned to the nearest centroid based on euclidean distance, forming k clusters. After assignment, the centroids are updated to be the mean of the points in their respective clusters. This process repeats until the centroids stabilize or a maximum number of iterations is reached.

It should be noted that the *normalized Euclidean distance* introduced in Equation 4.6 is used for distance calculations in K-means. This ensures that parameters with wider bounds do not dominate the clustering process, maintaining a balanced influence across all dimensions.

Local Movement

In each generation, the movement of every firefly is divided into two phases: a local movement phase within its assigned cluster and a global movement phase toward other clusters.

During the local movement phase, each firefly in cluster i considers only the other fireflies within the same cluster. If a neighboring firefly within the cluster has a higher intensity (i.e., a better objective function value), the firefly moves toward it according to the movement rule defined in Equation 4.2. Restricting interactions to within the cluster encourages localized exploration and helps refine solutions in promising regions of the search space.

Global Movement

In the global movement phase, each firefly considers only the centroids of other clusters as potential movement targets. If the centroid c_j of a cluster j has a higher intensity than the firefly's current position, the firefly moves toward that cluster. To account for the fact that each cluster is represented by a single centroid rather than multiple individual fireflies, the movement is scaled based on the cluster size n_c. This leads to the following updated movement rule:

$$x_i^{\text{new}} = x_i + \beta_0 e^{-\gamma\left(\frac{r}{\sqrt{n_c}}\right)^2}(c_j - x_i) + \alpha \cdot S(\text{rand} - \tfrac{1}{2}) \tag{4.10}$$

Here, the square root of the cluster size n_c is used to moderately increase the attraction toward larger clusters—representing a stronger collective signal—without overly amplifying the movement. Alternative scaling functions based on group size could also be considered depending on the specific optimization context.

Reduction of Cluster Size

Similar to the population control mechanism, the average size of the clusters is reduced over time to balance exploration and exploitation. The previously introduced *target fireflies per cluster*, denoted as n_{target}, is adapted dynamically at generation i according to:

$$n_{\text{target}}(i) = \max\left(n_{\text{target,min}}, \left\lfloor n_{\text{target,max}} \cdot \left(1 - \frac{i}{N_{\text{gen}}}\right)\right\rfloor\right), \tag{4.11}$$

where $n_{\text{target,max}}$ is the initial value, $n_{\text{target,min}}$ is the minimum allowed value (typically 1), and N_{gen} is the total number of generations.

This reduction in cluster size increases the resolution at which fireflies perceive other regions of the search space. Early in the optimization process, larger clusters span broader areas, providing a coarse assessment of each region's quality. As optimization progresses, clusters become smaller, allowing more fine-grained comparisons between regions. When combined with population control, this mechanism enhances convergence in later stages while avoiding excessive computational cost, since the total number of fireflies—and thus the number of function evaluations—also decreases over time.

4.3.1 Pseudocode

The pseudocode of the cFA is given in Algorithm 2.

Algorithm 2 Clustered Firefly Algorithm (cFA)

```
1: Initialize parameters: $\alpha$, $\gamma_{\min}$, $\gamma_{\max}$, $n_{\text{target,max}}$, population size $n$, etc.
2: Initialize $n$ fireflies randomly in the search space
3: for each generation $i = 1$ to $N_{\text{gen}}$ do
4:     Update $\gamma$: interpolate between $\gamma_{\min}$ and $\gamma_{\max}$
5:     Cluster fireflies using k-means
6:     Compute intensity for each cluster centroid
7:     for each firefly $x_i$ do
8:         Identify cluster $C_i$ to which $x_i$ belongs
9:         for each firefly $x_j \in C_i$ do
10:            if $f(x_j)$ better than $f(x_i)$ then
11:                Move $x_i$ toward $x_j$ using Equation 4.2
12:                Recompute $f(x_i)$
13:            end if
14:        end for
15:        for each centroid $c_j$ from other clusters do
16:            if $I(c_j)$ better than $f(x_i)$ then
17:                Move $x_i$ toward $c_j$ using Equation 4.10
18:                Recompute $f(x_i)$
19:            end if
20:        end for
21:    end for
22:    Reduce population size if applicable (remove worst fireflies)
23:    Reduce cluster size if applicable (decrease $n_{\text{target}}$)
24:    Update $\alpha$: $\alpha \leftarrow \alpha \cdot$ decay_factor
25: end for
26: Return the best firefly found
```

4.3.2 Time Complexity

The time complexity with respect to the number of objective function evaluations improves when using the cFA, as shown in the following. For simplicity, we ignore the effects of population reduction and dynamic cluster resizing.

In each generation, two types of movement contribute to the number of objective function calls: *local* and *global* movement.

Local movement Assuming that the average cluster size is n_{target}, then there are about n_{target}^2 potential movement updates for each cluster, as each firefly in a cluster compares itself with all other fireflies in its cluster. Since the total population is n and the average cluster size is n_{target}, the number of clusters is n/n_{target}, this results in the following number of objective function evaluations due to local movement:

$$n_{target}^2 \cdot n/n_{target} = n_{target} \cdot n$$

Global movement Each firefly may move toward the centroid of every other cluster, excluding its own, resulting in up to $(n/n_{\text{target}} - 1)$ movements per firefly. Thus, for a population of n fireflies, the total number of objective evaluations due to global movement is approximately:

$$n \cdot (n/n_{\text{target}}) = n^2/n_{\text{target}}$$

Total Combining both movement types, the total number of objective function evaluations per generation is:

$$n \cdot n_{\text{target}} + \frac{n^2}{n_{\text{target}}}$$

To enable a direct comparison with the original Firefly Algorithm, we assume $n_{\text{target}} = \sqrt{n}$. This assumption is reasonable when clustering a large population, as it ensures that the average cluster size remains significantly smaller than the total population size (i.e., $n_{\text{target}} \ll n$). Substituting this into the expression, we obtain:

$$n \cdot \sqrt{n} + \frac{n^2}{\sqrt{n}} = n^{3/2} + n^{3/2} = 2n^{3/2}$$

Thus, the total number of objective evaluations per generation becomes $O(n^{3/2})$, which is an improvement over the $O(n^2)$ evaluations required by the original FA (see Section 4.1).

4.3.3 Firefly Visualization

In this section, a simple optimization problem is visualized to provide insight into how the cFA operates.

As an objective function, the *Schwefel* function is used [33], which, for a d-dimensional problem, is defined as follows:

$$f(x) = -\left(418.9829 \cdot d - \sum_{i=1}^{d} x_i \sin\left(\sqrt{|x_i|}\right)\right). \tag{4.12}$$

Note, that the function was negated to transform it into a maximization problem, aligning with the FA's preference for brighter (higher fitness) solutions.
The Schwefel function is typically defined over $x_i \in [-500, 500]$ for all $i = 1, \ldots, d$, with the global maximum located at $x_i = 420.9687$ for all i. The corresponding maximum function value is 0.

This function has many local optima, making it well suited to demonstrate the ability of the cFA to navigate complex multimodal search spaces. A 3D plot illustrating the rugged landscape of the 2-dimensional Schwefel function is shown in Figure 4.1. The abundance of local optima increases the risk of premature convergence, as optimization algorithms may easily become trapped in suboptimal regions. Additionally, the global optimum is located far from the center of the search space, further challenging algorithms that tend to favor central or local exploration strategies.

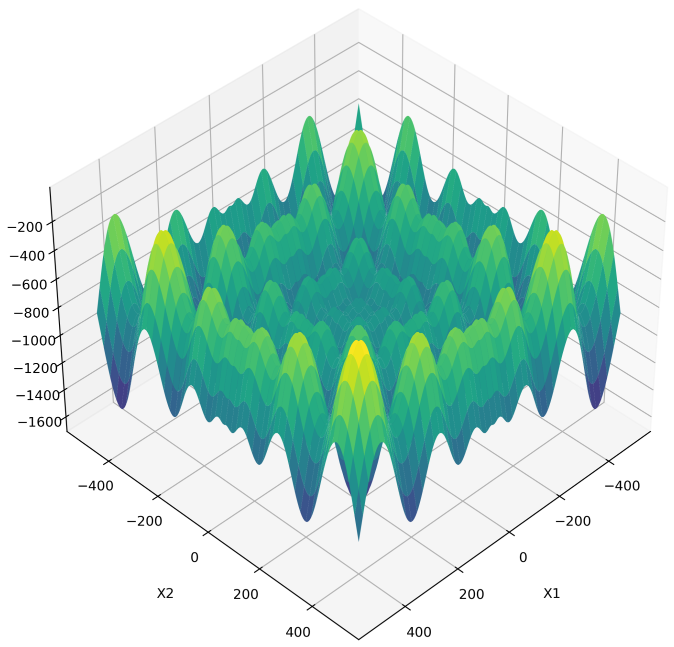


Figure 4.1: 3D plot of the Schwefel function landscape used for visualization.

4.3.4 Setup

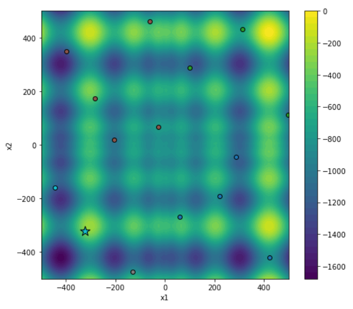
For the visualization, the cFA was run on the two-dimensional Schwefel function over 15 generations. The initial population consisted of 15 fireflies randomly distributed across the search space $x_i \in [-500, 500]$. The randomization parameter α was initialized at 0.5 and decayed by a factor of 0.95 each generation. The attractiveness parameter γ was varied linearly from 1 to 7 over the generations.
An average target cluster size value of 3 was chosen with a minimum target size of 1. Population control was turned off for this experiment to better show the swarming behavior of the fireflies.

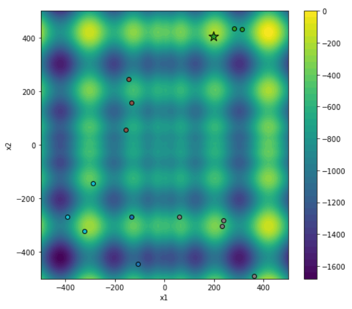
4.3.5 Results

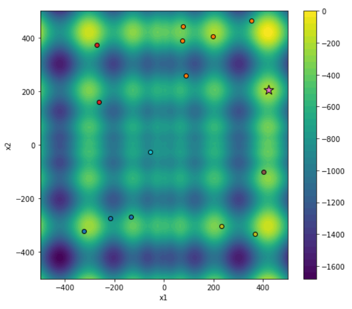
Figure 4.2 illustrates the evolution of the firefly population over 15 generations. Initially (Generation 4.2a), the fireflies are widely dispersed across the search space. As the optimization proceeds, they progressively explore the landscape and begin to cluster toward regions of higher objective value. By Generation 4.2i, the majority of fireflies have swarmed near the global optimum, demonstrating the algorithm's ability to balance exploration and exploitation over time.

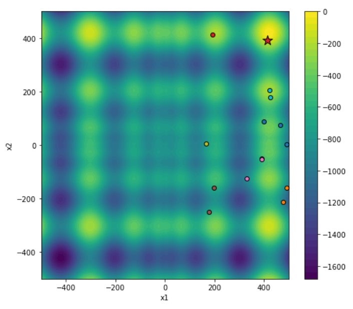
The best fireflies, depicted by star symbols, were initially located at different local optima but eventually converged to the global optimum. This behavior showcases the cFA's capacity to perform global search and overcome attraction to local optima.

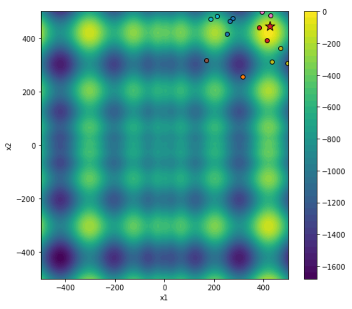
It should be noted that this optimization run was specifically selected to illustrate the desired behavior. Due to the stochastic nature of the algorithm, convergence to the global optimum is not guaranteed. In other runs with the same experimental setup, the fireflies occasionally became trapped in local optima. Therefore it is important to choose the right amount of fireflies—to balance computational efficiency and solution quality.

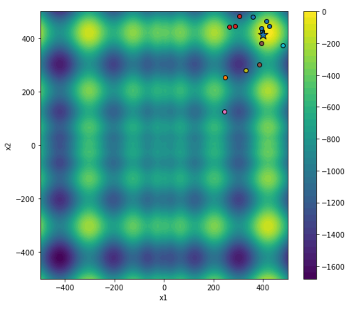

(a) Generation 0

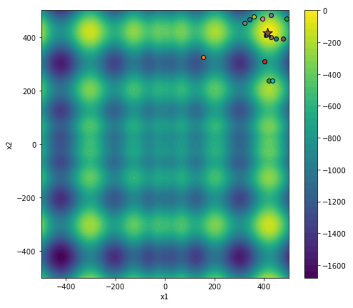

(b) Generation 1

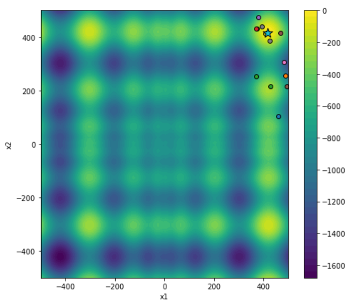

(c) Generation 2


(d) Generation 3


(e) Generation 4


(f) Generation 5


(g) Generation 9


(h) Generation 12

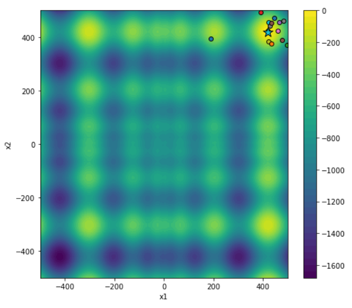

(i) Generation 15

Figure 4.2: Evolution of the firefly population over generations on the Schwefel function. The best firefly in each generation is highlighted with a star symbol. The color of a firefly indicates to which cluster that firefly belongs.

Chapter 5

Benchmark Experiments

To evaluate the effectiveness of the proposed cFA, its performance is benchmarked against the original FA, a modified FA that includes all non-cluster related modifications and a baseline grid search approach on several standard test functions.

As a performance measure, the best-so-far function value is plotted against the number of objective function evaluations. This metric is chosen because evaluating the objective function is typically the most computationally expensive operation during the optimization process.

Formally, the *best-so-far* value after t evaluations is defined as

$$f_t^* = \max\{f(x_i) \mid 1 \leq i \leq t\}, \tag{5.1}$$

where $f(x_i)$ denotes the objective function value at the i-th evaluation.

If the objective function values span several orders of magnitude, a logarithmic transformation is applied to better highlight improvements near the optimum. The transformed performance metric is defined as

$$y_t = -\log_{10}(|f_t^*| + \varepsilon), \tag{5.2}$$

where ε is a small constant added to avoid numerical instability near zero.

5.1 Benchmark Functions

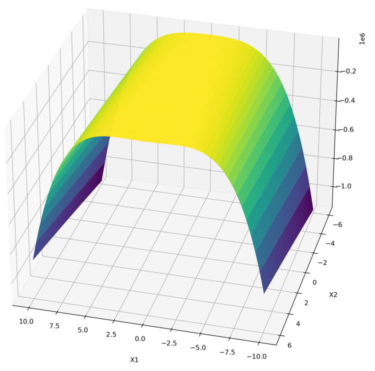
Three functions were selected to evaluate the performance of the algorithms. It should be noted that the sign of each function has been inverted from its original definition, converting the problems into maximization tasks to align with the FA's metaphor of attraction toward brighter (i.e., higher-scoring) individuals.

The first function is the *Schwefel* function (Equation 4.12), which was previously introduced in Section 4.3.3. It is characterized by a rugged landscape with many local maxima and deceptive regions, making it well-suited to assess the algorithm's global exploration capabilities.

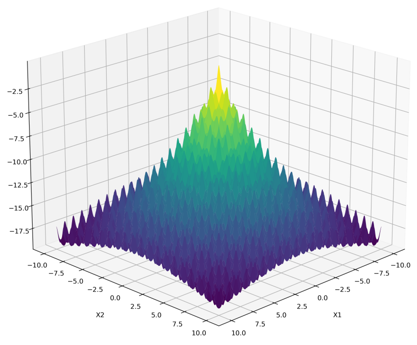
The second benchmark is the *Rosenbrock* function [33], mathematically defined as:

$$\text{rosenbrock}(x) = -\sum_{i=1}^{d-1} \left[100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2\right] \tag{5.3}$$

In contrast to the *Schwefel* function, the (negated) *Rosenbrock* function is unimodal, with its global maximum situated along a long, narrow, curved ridge. While the general region of the optimum is relatively easy to locate, the ridge's flatness and curvature make precise convergence particularly challenging. As such, this function emphasizes the importance of convergence mechanisms over broad exploration strategies. The global maximum of the d-dimensional Rosenbrock function is located at $x_i = 1$ for all $i \in \{1, ..., d\}$, with a corresponding function value of 0. A 3D plot of the 2-dimensional function is shown in Figure 5.1a.



(a) Rosenbrock function



(b) Ackley function

Figure 5.1: 3D plots of the 2-dimensional Rosenbrock and Ackley benchmark functions.

Lastly, the *Ackley* function [33] was chosen, which is mathematically defined as

$$\text{ackley}(x) = -\left(-a\exp\left(-b\sqrt{\frac{1}{d}\sum_{i=1}^{d} x_i^2}\right) - \exp\left(\frac{1}{d}\sum_{i=1}^{d}\cos(cx_i)\right) + a + \exp(1)\right) \tag{5.4}$$

This function features many local maxima scattered across a relatively flat landscape surrounding a large global optimum at the center. To reach the optimum, an algorithm must both explore broadly to escape local optima and converge precisely to avoid over-shooting. As such, this benchmark is well suited to highlight algorithms that effectively balance exploration and convergence. The global maximum of the Ackley function lies at the center, with $x_i = 0$ for all $i \in \{1, \ldots, d\}$, corresponding to a function value of 0. Standard values are chosen for parameters: $a = 20$, $b = 0.2$ and $c = 2\pi$.

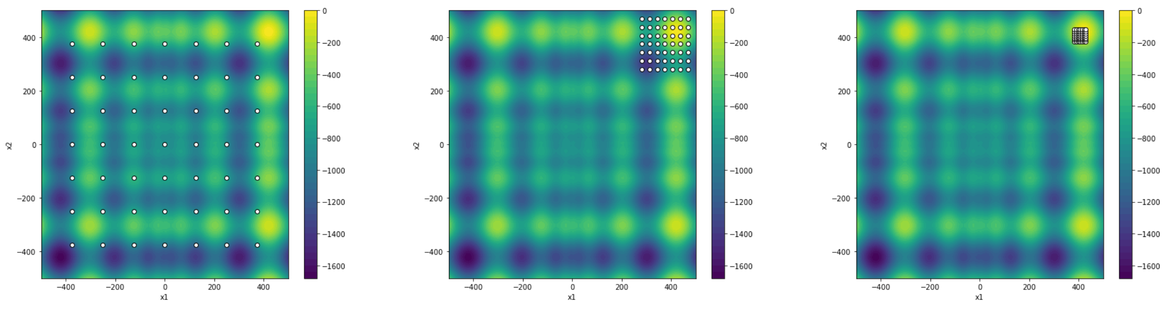
In Figure 5.1b, the landscape of the two-dimensional Ackley function is visualized.

5.2 Grid Search

Grid Search is an exhaustive search optimization technique that evaluates the objective function at a fixed number n of regularly spaced points across the search space. While it can perform well in low-dimensional spaces, it suffers from the *curse of dimensionality* [34], meaning that the number of points N_{grid} to be evaluated grows exponentially with the number of dimensions d:

$$N_{\text{grid}} = n^d \tag{5.5}$$

In these baseline experiments, a modified version of Grid Search is used that iteratively *zooms in* on the best solution found so far. In each iteration, a new, smaller grid is constructed around the previously best-performing point, allowing the algorithm to refine its estimate over time. A visualization of this process for the 2-dimensional Schwefel function is shown in Figure 5.2.



(a) Iteration 0 **(b)** Iteration 1 **(c)** Iteration 2

Figure 5.2: Evolution of the grid search process over iterations on the 2-D Schwefel function.

5.3 Experimental Setup

Based on the expected difficulty of the optimization problems—favoring either convergence or exploration—different hyperparameters were selected for each algorithm. For each benchmark problem, a learning budget was defined as the maximum number of objective function evaluations.

To ensure a fair comparison, the original and modified FAs were initialized with the same number of fireflies. However, since fireflies in the cFA move less frequently due to group-based movement, the number of initial fireflies was increased when necessary to ensure the algorithm fully utilized the available learning budget.

Fixed values of $\beta_0 = 1$ and $\alpha = 0.5$ were used in Equation 4.2 for all Firefly-based algorithms.

The problem-specific hyperparameters used in the experiments are summarized in Tables 5.1, 5.2, and 5.3.

For the Grid Search algorithm, the number of evaluation points per dimension and the number of zoom-in iterations were selected based on the dimensionality of the optimization problem. These settings are also listed in Tables 5.1, 5.2, and 5.3. Here, it should be noted that the points per dimension and the number of zoom-in iterations are smaller in the higher-dimensional experiments due to the previously mentioned *curse of dimensionality*.

As mentioned earlier, the best-so-far value, defined in Equation 5.1, was plotted against the number of objective function evaluations. For the Firefly Algorithms, the mean best-so-far value over 100 independent runs was used to observe the general performance, as the algorithms are inherently stochastic. Since Grid Search is deterministic, only a single run was performed.

For the *Schwefel* optimization problem, experiments were conducted in dimensions 5 and 10, with budgets of 20,000 and 40,000 objective function evaluations, respectively. The function was evaluated over the hypercube $x_i \in [-500, 500]$ for all $i \in \{1, \dots, d\}$. The chosen hyperparameters are summarized in Table 5.1.

Parameter	**Modified Firefly**	**Original Firefly**	**Grid Search**	**Cluster Firefly**
Initial number of fireflies ($n_{\max}$)	70	70	–	70
Number of generations	100	100	–	100
Minimum number of fireflies ($n_{\min}$)	30	–	–	30
α (randomness weight)	0.5, decay factor 0.97	0.5	–	0.5, decay factor 0.97
γ (attractiveness decay)	Linear $1 \to 3$	1	–	Linear $1 \to 3$
Grid resolution per dimension (n)	–	–	5 (2)	–
Number of zoom-in iterations	–	–	10 (40)	–
Target fireflies per cluster	–	–	–	5

Table 5.1: Hyperparameters used for the *Schwefel* benchmark problem.

Experiments on the *Rosenbrock* function were conducted in dimensions 5 and 15 over the hypercube $x_i \in [-50, 50]$ for all $i \in \{1, \dots, d\}$. Since convergence generally takes longer due to the flatness around the maximum, a learning budget of 100,000 objective function evaluations was used. Due to the steepness of the function, the logarithmic transformation defined in Equation 5.2 was applied to the best-so-far values, using $\varepsilon = 10^{-8}$ to avoid numerical issues near zero. The hyperparameters for the experiments are summarized in Table 5.2.

Parameter	**Modified Firefly**	**Original Firefly**	**Grid Search**	**Cluster Firefly**
Initial number of fireflies ($n_{\max}$)	60	60	–	80
Number of generations	150	150	–	350
Minimum number of fireflies ($n_{\min}$)	20	–	–	20
α (randomness weight)	0.5, decay factor 0.95	0.5	–	0.5, decay factor 0.95
γ (attractiveness decay)	Linear 0.1$\rightarrow$ 10	1	–	Linear 0.1$\rightarrow$ 10
Grid resolution per dimension (n)	–	–	7 (2)	–
Number of zoom-in iterations	–	–	6 (4)	–
Target fireflies per cluster	–	–	–	8

Table 5.2: Hyperparameters used for the *Rosenbrock* benchmark problem.

Lastly, for the *Ackley* optimization problem, experiments in dimension 5 and 15 were conducted on the hypercube $x_i \in [-32.768, 32.768]$ for all $i \in \{1 \dots d\}$. Again, a learning budget of 100,000 objective function evaluations was used. Table 5.3 summarizes the hyperparameters of the experiment.

Parameter	**Modified Firefly**	**Original Firefly**	**Grid Search**	**Cluster Firefly**
Initial number of fireflies ($n_{\max}$)	60	60	–	80
Number of generations	125	125	–	350
Minimum number of fireflies ($n_{\min}$)	25	–	–	25
α (randomness weight)	0.5, decay factor 0.95	0.5	–	0.5, decay factor 0.95
γ (attractiveness decay)	Linear 0.1 $\rightarrow$ 10	0.1	–	Linear 0.1 $\rightarrow$ 10
Grid resolution per dimension (n)	–	–	7 (2)	–
Number of zoom-in iterations	–	–	6 (4)	–
Target fireflies per cluster	–	–	–	8

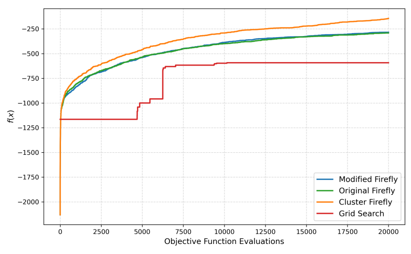
Table 5.3: Hyperparameters used for the *Ackley* benchmark problem.

5.3.1 Hardware Specifications

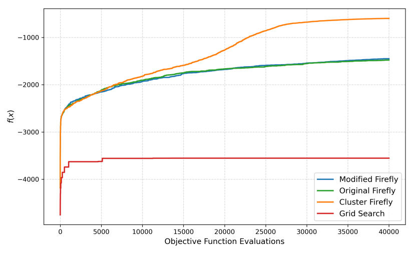
All experiments were conducted on an Acer Nitro AN515-54 laptop running Windows 10 Home. The system was equipped with an Intel Core i7-9750H processor (6 cores, 12 threads, 2.6 GHz base frequency) and 16 GB of RAM. All computations were executed on the CPU.

5.4 Results

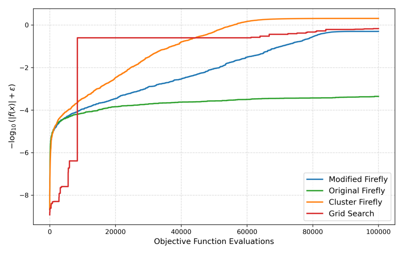
Figure 5.3 shows the convergence behavior of the three algorithms on the different benchmarking problems.



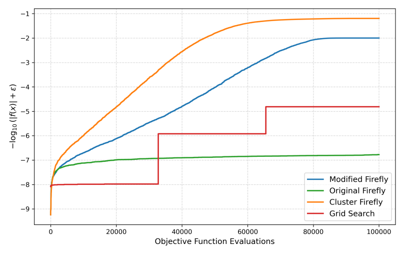
(a) Schwefel, 5D



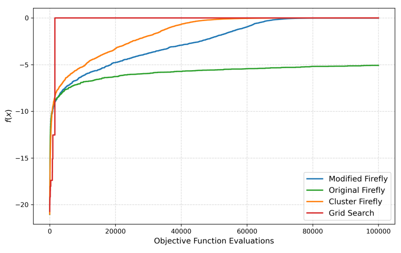
(b) Schwefel, 10D



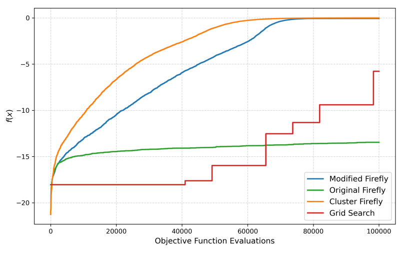
(c) Rosenbrock, 5D



(d) Rosenbrock, 15D



(e) Ackley, 5D



(f) Ackley, 15D

Figure 5.3: Convergence behavior of the three algorithms across different benchmark problems and dimensions.

5.4.1 Results on the Schwefel Function

Figures 5.3a and 5.3b show the performance on the 5- and 10-dimensional Schwefel functions. The best performance is achieved by the cFA. The performance of the modified and original FA is similar across both problems, with the modified version performing slightly better. All Firefly-based algorithms perform significantly better than Grid Search for both dimensions. It should be noted that for both dimensions, the global maximum was not found (on average) within the experimental budget.

5.4.2 Results on the Rosenbrock Function

The performance on the 5- and 15-dimensional *Rosenbrock* functions is shown in Figures 5.3c and 5.3d.
In the 5-dimensional case, Grid Search achieves the best performance in the early stages, finding high-valued solutions more quickly. All Firefly-based algorithms show similar behavior in the early stages; however, the original FA stagnates and ultimately yields the worst results. In contrast, both the modified and clustered FAs continue to improve, with the clustered variant converging faster and ultimately reaching a better objective value than Grid Search.

On the 15-dimensional *Rosenbrock* function, both the modified and clustered FAs outperform the original version and Grid Search.
The clustered variant demonstrates the fastest convergence and consistently yields the best solutions, while the modified variant performs second best, converging to slightly less optimal solutions on average. The original FA remains the weakest performer. As with the *Schwefel* function experiments, on average, the exact global optimum was not reached within the allocated budget.

5.4.3 Results on the Ackley Function

The results for the *Ackley* function are shown in Figures 5.3e and 5.3f.
As in the previous experiment, Grid Search performs well on the 5-dimensional problem—this time achieving the best overall performance, followed by the clustered Firefly Algorithm and the modified variant.
For the 15-dimensional case, the cFA achieves the best performance, once again demonstrating faster convergence than the modified version. In both experiments, the original FA exhibits the weakest performance.
Unlike in the previous benchmarks, the global optimum is reached: by Grid Search in the 5-dimensional case, and by both the clustered and modified Firefly Algorithms in the 5- and 15-dimensional cases.

5.5 Discussion

Overall, the cFA performed the best, demonstrating both the ability to explore the search space and to converge more rapidly.

The original FA exhibited weaknesses in convergence. However, it performed with a similar performance in the Schwefel problem as the modified algorithm. A possible explanation is that the Schwefel landscape is extremely challenging, characterized by many local optima. In such cases, broad exploration is crucial, and the original FA's strong exploratory behavior becomes advantageous.

The Grid Search algorithm showed solid performance. As expected, it performed worse on higher-dimensional problems but still outperformed the original FA on the Rosenbrock and Ackley functions. It is worth noting that Grid Search likely benefited from the symmetrical nature of these two benchmark functions, where the global optimum is located near the center of the search space. In contrast, the Schwefel function's global optimum lies far from the center; once Grid Search zooms in on a local optimum, it cannot recover, leading to poor performance.

The modifications introduced to the FA–including dynamic population control, adaptive attractiveness parameters, and randomness decay–appear to improve the balance between exploration and exploitation. These adjustments enable the algorithm to explore the search space broadly in the early stages while progressively focusing on convergence toward the best–found solutions later on, particularly in high–dimensional and multimodal landscapes.

The clustering mechanism, which combines local search within a cluster and guided movement toward other clusters, appears to improve both global exploration and convergence. This hybrid approach allows fireflies to first exploit local optima and then migrate to stronger "colonies" when their own cluster appears suboptimal. One possible explanation for the strong performance is a smoothing effect introduced by using cluster centroids. Rather than reacting to noisy or highly localized variations in the fitness landscape, fireflies are influenced by the aggregated behavior of nearby solutions, guiding them toward broadly promising regions of the search space. This may help avoid premature convergence while still maintaining directional focus.

Chapter 6

Parameter Estimation of Theoretical ASM1 Wastewater Treatment Plant

Wastewater treatment plants (WWTPs) play a crucial role in protecting public and environmental health by removing pathogens, pollutants, and chemical substances from wastewater. To achieve this, WWTPs employ a combination of physical, biological, and chemical processes. As wastewater is generated by both the public sector (e.g., municipalities) and the private sector (e.g., industrial sources), WWTPs are indispensable in modern society.

Despite their necessity, WWTPs pose a significant challenge: high energy consumption. It is estimated that the water sector accounts for approximately 4% of global energy usage, with WWTPs contributing around 1% to this total [12]. Consequently, improving the energy efficiency of WWTP operations while maintaining or enhancing effluent quality is a matter of great concern.

In the previous semester, our group worked on a dynamical simulation model of a WWTP intended for integration with MPC, with the aim of deriving near-optimal control strategies to reduce energy consumption and improve water quality. This semester, the group has been divided into two subgroups. While one focuses on developing MPC-based control strategies, this project concentrates on parameter estimation for WWTP models in this use-case. Accurate parameter estimation is essential for ensuring that the dynamical simulation model reliably reflects the actual system, which is a prerequisite for deriving effective control strategies.

Case Study This use case was conducted in collaboration with Dansk Hydraulisk Institut (DHI)[1], an independent research and consulting institute focused on the water environment. DHI proposed several topics of interest, one of which centered on parameter estimation. The case study investigated in this thesis is based on the following problem formulation provided by DHI:

[1]`https://www.dhigroup.com/`

Process model re-calibration: Mechanistic/deterministic models, such as WEST or MIKE[2], are comprised of many different parameters and equations for modelling biological, chemical, and physical processes. The accuracy and reliability of these models directly depend on correctly calibrating the input parameters. However, even if models are correctly calibrated using data from a specific period, the accuracy of the model going forward can deteriorate. Therefore, to keep the model accurate, regular recalibration of parameters may be required. This process of recalibration can be time-consuming and often relies on human engineers to manually update the parameters.
This topic will investigate whether it is possible to use data-driven approaches to recalibrate model parameters within mechanistic models.

In the following experiments, the clustered Firefly Algorithm is employed to estimate kinetic parameters in a simulated wastewater treatment plant based on the ASM1 model. To evaluate the generalization capability of the estimated parameter set, the model's performance is also tested on a separate timeframe that was not used during the parameter estimation process.

The following section provides an overview of the plant layout and model formulation, including the differential equations and parameters used. For a more detailed discussion of the biological processes and underlying assumptions, the reader is referred to the semester project report that preceded this thesis [35].

6.1 Model Description

Plant Layout

A simple denitrifying wastewater treatment plant layout was chosen for this experiment. The layout is inspired by a standard configuration available in the modeling software WEST, developed by DHI. It consists of two **biological process tanks**, a **secondary settling tank**, and internal **recycle flows**.

Wastewater from the source (e.g., a municipality) enters the system as **influent** and first flows into the **anoxic tank**, where no oxygen is added. In this tank, **nitrate** (NO_3^-) is reduced to nitrogen gas by heterotrophic bacteria through denitrification, effectively removing nitrogen from the wastewater.

The flow then continues to the **aerobic tank**, where oxygen is supplied through aeration equipment. In this tank, heterotrophic bacteria oxidize organic matter, and autotrophic bacteria convert **ammonium** (NH_4^+) into nitrate via nitrification. To support continued denitrification, a portion of the nitrate-rich mixed liquor is recycled back to the anoxic tank via an **internal recycle** line.

[2]MIKE and WEST are software tools used for modeling water and environmental systems. See dhigroup.com for more information.

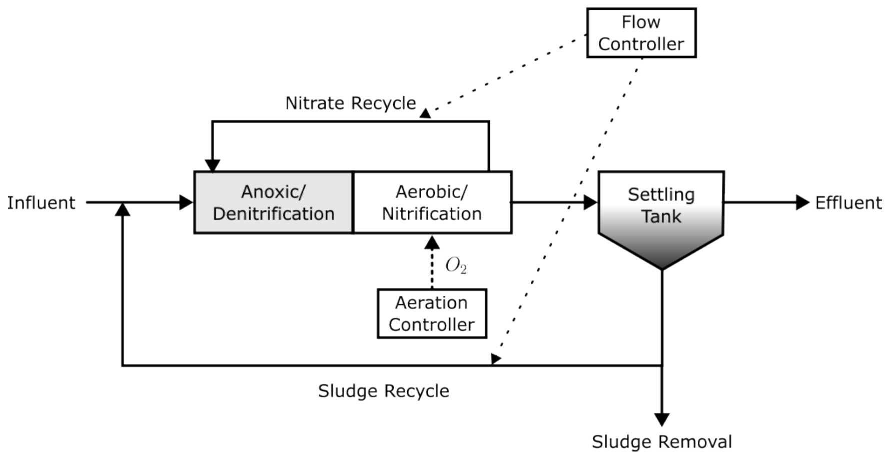


Figure 6.1: Waste Water Treatment Plant Layout.

The remaining flow is sent to the **secondary settling tank** (clarifier), where particulate biomass settles by gravity. The settler has two outputs: the **overflow** (effluent), which contains mainly soluble compounds and exits the system, and the **underflow**, which contains concentrated biomass. This underflow is recycled to the anoxic tank as **return activated sludge (RAS)** to maintain sufficient biomass concentrations in the biological reactors. Excess sludge that is not recycled is removed and typically undergoes further treatment in dedicated sludge processing units.

An overview of the plant layout is shown in Figure 6.1.

Biological Tank Model Description

The *Activated Sludge Model No. 1* (ASM1) [13] is a mechanistic model that describes the temporal evolution of soluble and particulate components in a WWTP through a system of ordinary differential equations (ODEs). It is widely used for simulating the biological processes of carbon oxidation, nitrification, and denitrification.

Table 6.1 lists the main state variables (components) used in ASM1.

Component	**Notation**
Readily biodegradable substrate	S_S
Slowly biodegradable substrate	X_S
Soluble inert organic matter	S_I
Particulate inert organic matter	X_I
Heterotrophic biomass	$X_{B,H}$
Autotrophic biomass	$X_{B,A}$
Particulate products arising from biomass decay	X_P
NH_4^+ and NH_3 ammonia nitrogen	S_{NH}
Soluble biodegradable organic nitrogen	S_{ND}
Particulate biodegradable organic nitrogen	X_{ND}
Nitrate and nitrite nitrogen	S_{NO}
Alkalinity	S_{ALK}
Oxygen	S_O

Table 6.1: List of ASM1 components. Soluble components are indicated by S_x and particulate components by X_x

Model Parameters

ASM1 includes a set of kinetic and stoichiometric parameters that define the behavior of each process. Table 6.2 provides an overview of these parameters along with standard values for each parameter. The standard values are taken from [13].

Description	Symbol	Value (Units)
Stoichiometric Parameters		
Heterotrophic yield	Y_H	0.67 (g COD/g COD)
Autotrophic yield	Y_A	0.24 (g COD/g N)
Fraction of biomass yielding particulate products	f_p	0.08 (-)
Mass N / mass COD in biomass	i_{XB}	0.086 (g N/g COD)
Mass N / mass COD in products from biomass	i_{XP}	0.06 (g N/g COD)
Kinetic Parameters		
Heterotrophic maximum specific growth rate	μ_H	6.0 (d^{-1})
Heterotrophic decay rate	b_H	0.62 (d^{-1})
Half-saturation coefficient for heterotrophs	K_S	20 (g COD/m^3)
Oxygen half-saturation coefficient for heterotrophs	K_{OH}	0.20 (g O_2/m^3)
Nitrate half-saturation coefficient for denitrifying heterotrophs	K_{NO}	0.50 (g N/m^3)
Autotrophic maximum specific growth rate	μ_A	0.80 (d^{-1})
Autotrophic decay rate	b_A	0.15 (d^{-1})
Oxygen half-saturation coefficient for autotrophs	K_{OA}	0.4 (g O_2/m^3)
Ammonium half-saturation coefficient for autotrophs	K_{NH}	1.0 (g N/m^3)
Correction factor for anoxic growth of heterotrophs	η_g	0.80 (-)
Ammonification rate	k_a	0.08 m^3 g COD d^-1
Maximum specific hydrolysis rate	k_h	3.0 (d^{-1})
Half-saturation coefficient for hydrolysis of slowly biodegradable substrate	K_X	0.03 (g COD/g COD)
Correction factor for anoxic hydrolysis	η_h	0.40 (-)

Table 6.2: Stoichiometric and kinetic parameters used in the ASM1 model at 20°C. Units follow standard ASM1 conventions.

Process Rates

The ASM1 model defines eight biological and chemical processes that govern the dynamic evolution of the interdependent state variables listed in Table 6.1. Each process is associated with a process rate expression, typically based on *Monod kinetics* and stoichiometric coefficients. Table 6.3 summarizes these processes and their corresponding rate expressions. Note that these processes use the parameters introduced in Table 6.2.

Table 6.3: ASM1 processes and their rate expressions

Process	**Rate Expression**
Aerobic growth of heterotrophs (ρ_1)	$\mu_H \left(\frac{S_S}{K_S+S_S}\right)\left(\frac{S_O}{K_{O,H}+S_O}\right) X_{B,H}$
Anoxic growth of heterotrophs (ρ_2)	$\mu_H \left(\frac{S_S}{K_S+S_S}\right)\left(\frac{K_{O,H}}{K_{O,H}+S_O}\right)\left(\frac{S_{NO}}{K_{NO}+S_{NO}}\right) \eta_g X_{B,H}$
Aerobic growth of autotrophs (ρ_3)	$\mu_A \left(\frac{S_{NH}}{K_{NH}+S_{NH}}\right)\left(\frac{S_O}{K_{O,A}+S_O}\right) X_{B,A}$
Decay of heterotrophs (ρ_4)	$b_H X_{B,H}$
Decay of autotrophs (ρ_5)	$b_A X_{B,A}$
Ammonification of soluble organic nitrogen (ρ_6)	$k_a S_{ND} X_{B,H}$
Hydrolysis of entrapped organics (ρ_7)	$k_h \frac{X_S/X_{B,H}}{K_X+(X_S/X_{B,H})}\left[\left(\frac{S_O}{K_{OH}+S_O}\right) + \eta_b \left(\frac{K_{OH}}{K_{OH}+S_O}\right)\left(\frac{S_{NO}}{K_{NO}+S_{NO}}\right)\right] X_{B,H}$
Hydrolysis of entrapped organic nitrogen (ρ_8)	$\rho_7 \left(\frac{X_{ND}}{X_S}\right)$

Each state variable in ASM1 evolves over time according to its associated processes and their stoichiometric contributions. In general, the rate of change r_Y for component Y is given by some weighted sum of the eight processes as follows:

$$r_Y = \sum_{i=1}^{8} v_{iY} \rho_i, \tag{6.1}$$

where ρ_i denotes the rate of process i, and v_{iY} is the stoichiometric coefficient that defines the impact of process i on component Y. The sign of v_{iY} indicates whether the component is produced (positive) or consumed (negative) by the process.

To illustrate how process rates affect individual state variables, consider the component S_S, which is influenced by processes ρ_1, ρ_2, and ρ_7. The biological rate of change for S_S is given by:

$$\frac{dS_S}{dt} = -\frac{1}{Y_H} \cdot \rho_1 - \frac{1}{Y_H} \cdot \rho_2 + 1 \cdot \rho_7, \tag{6.2}$$

where Y_H is the heterotrophic yield coefficient, a stoichiometric parameter. As indicated by the signs, S_S is consumed in processes ρ_1 and ρ_2, and produced in process ρ_7.

A compact notation for this system of equations is the Gujer matrix [36], which arranges the stoichiometric coefficients of all components and processes into a matrix. This matrix can then be used together with the process rate vector to compute the rate of change of each component according to Equation 6.1. The Gujer matrix for ASM1 is provided in Appendix A.1.

Ultimately, the concentrations of the various components (represented by the state variables) evolve according to a set of simultaneous differential equations defined by these process rates.

Settling Model

The settling and recycling of particulate solids is an essential process in the secondary treatment stage of WWTPs. To simulate sludge settling, we include a one-dimensional clarifier model based on the work of *Takács et al.* [37]. The model divides the settler into multiple vertical layers and describes solids transport as a combination of gravity-driven settling and bulk flow convection due to influent, underflow, and overflow. Flux limitations are applied to ensure physically realistic transport, especially in zones of high solids concentration. The model accounts for settling phenomena such as hindered settling, compression, and sludge blanket formation.

In this experiment, we use a 10-layer configuration with a middle-layer influent feed, similar to the implementation described in the preceding semester report [35]. For further details on the mathematical formulation and assumptions of the Takács model, the reader is referred to that report.

Transport and Flow Terms

In a complete plant model, the change in concentration of each component is influenced not only by biological and chemical transformation rates (as defined in ASM1), but also by advective transport due to water flows between units. The general mass balance for a component Y_j in a given tank is:

$$\frac{dY_j}{dt} = r_{Y_j} + \sum_{\text{inlets}} Q_{\text{in}} \cdot \left(Y_j^{\text{in}} - Y_j \right) \tag{6.3}$$

where:

- Q_{in} is the volumetric flow rate into the tank (m^3/d). In this model, the outflow from each unit is assumed to equal the inflow,
- Y_j^{in} is the concentration of component Y_j in the incoming stream,
- Y_j is the current concentration in the tank,
- r_{Y_j} is the biological transformation rate from Equation 6.1.

This equation applies to all flow streams, including the influent, internal recycles, effluent, and underflow/overflow from the settler. These transport terms are essential for coupling the ASM1 biological processes with the hydraulic dynamics of the plant.

Aeration Control via PI Controller

The dissolved oxygen concentration in the aerobic tank is regulated using a Proportional-Integral (PI) controller [38]. The controller dynamically adjusts the oxygen input rate to maintain the concentration of dissolved oxygen S_O close to a predefined setpoint $S_{O,\text{set}}$.

The PI controller computes the control error as:

$$e(t) = S_{O,\text{set}} - S_O(t) \tag{6.4}$$

The aeration input is then calculated as:

$$u(t) = K_P \cdot e(t) + K_I \cdot \int_0^t e(\tau)\, d\tau \tag{6.5}$$

where:

- K_P is the proportional gain,
- $K_I = \frac{K_P}{T_i}$ is the integral gain, with T_i being the integral time constant,
- $e(t)$ is the error between the desired and actual oxygen concentration.

The resulting aeration rate $u(t)$ is applied as an input to the oxygen mass balance in the aerobic tank, directly influencing the dynamics of S_O.

Controller

In real-world WWTPs, operators can adjust control variables such as the dissolved oxygen setpoint and the recycle flow rates (return sludge and internal nitrate recycle) to meet specific objectives, such as optimizing energy efficiency or effluent quality. However, since this thesis does not focus on operational optimization, these control variables are held constant throughout the simulation. The values used in this model are summarized in Table 6.4.

Control Variable	**Value**
Dissolved oxygen setpoint	1.5 mg/L
Underflow (return sludge) recycle rate	14 m^3/min
Nitrate liquor (internal) recycle rate	38.5 m^3/min

Table 6.4: Fixed values of control variables during simulation.

6.2 Experimental Setup

6.2.1 Modelling Software

The wastewater treatment plant was modeled using the UPPAAL software suite, which is designed for modeling, simulation, and verification of stochastic hybrid systems through timed automata [39]. With the addition of UPPAAL Stratego, the tool also supports the synthesis of optimal control strategies [7].

In the context of this project, the primary use of UPPAAL was for simulation purposes—specifically, to evaluate the behavior of the WWTP model under different parameter configurations. For a more detailed explanation of the modeling process and the use of UPPAAL, the reader is again referred to the previous semester's report [35].

Alternatively, other simulation tools such as WEST could have been used to simulate the WWTP. However, due to prior familiarity with UPPAAL and the fact that its performance—both in terms of speed and accuracy—is comparable to WEST for this specific plant layout (see previous semester report), UPPAAL was chosen.

6.2.2 Parameter Setting

The experimental setup and selection of sensitive parameters in this study follow the approach used by *Du et al.* [28], who based their work on established sensitivity analyses of the ASM1 model.
A set of seven highly sensitive parameters and their respective ranges is adopted, as shown in Table 6.5. The remaining, less sensitive parameters are fixed to the standard values defined by Henze et al. [13] that have been shown in Table 6.2.

Sensitive Parameters		
Description	**Symbol**	**Typical Range**
Heterotrophic yield	Y_H	0.38 – 0.75 (g COD/g COD)
Heterotrophic decay rate	b_H	0.09 – 4.38 (d^{-1})
Heterotrophic maximum specific growth rate	μ_H	3 – 13.3 (d^{-1})
Autotrophic maximum specific growth rate	μ_A	0.34 – 0.65 (d^{-1})
Oxygen half-saturation coefficient for autotrophs	K_{OA}	0.5 – 2.0 (g O_2/m^3)
Ammonium half-saturation coefficient for autotrophs	K_{NH}	0.6 – 3.6 (g N/m^3)
Half-saturation coefficient for heterotrophs	K_S	10 – 180 (g COD/m^3)

Table 6.5: Seven sensitive ASM1 parameters and their typical value ranges used in this study.

6.2.3 Ground Truth Dataset

Since no real-life measurement data was available for this project, a synthetic dataset was generated for the purpose of parameter estimation based on the ASM1 treatment plant.

To initialize the model's internal state, a steady-state inflow was applied for a period of 10,000 minutes. This approach is commonly used in the field of wastewater treatment to ensure that the simulated system reaches a stable operating condition before introducing dynamic inflow variations. After initialization, a dynamic inflow pattern obtained from the WEST modeling software was used to simulate realistic influent fluctuations.

Following the approach in [28], four effluent components were selected as ground truth targets: S_S, X_S, S_{NH}, and S_{NO}. While these components were experimentally measured in their study, here they are generated by simulating the model with a known parameter configuration.

To produce the ground truth dataset, the more sensitive parameters were randomly selected within the ranges listed in Table 6.5, while all other parameters were fixed to their default values:

Y_H	b_H	μ_H	μ_A	K_{OA}	K_{NH}	K_S
0.69	3.34	6.16	0.51	1.37	2.79	38.52

Table 6.6: Parameter values used to generate the ground truth dataset.

With this configuration, the model was simulated under dynamic inflow conditions for a period of 3 days, using a fixed timestep of 10 minutes. The resulting time series of the selected effluent components serve as the ground truth for the parameter estimation experiment.

6.2.4 Objective Function

To apply the cFA for parameter estimation, an objective function is required to evaluate the quality of each firefly (each candidate parameter configuration).

Since simulated effluent data is available over a 3-day period, the objective function is based on the discrepancy between the ground truth time series and the output of a simulation using the candidate parameters. More specifically, the negative **mean squared error** (MSE) is used to quantify this difference.

To prevent components with larger numeric ranges from dominating the error, all time series are normalized using min-max normalization based on the ground truth values. Let $y_{i,t}^{\text{gt}}$ denote the normalized ground truth value of component i at time t, and $y_{i,t}^{\text{sim}}$ the corresponding normalized simulation value. Then the MSE-based objective function is defined as:

$$\text{MSE} = -\frac{1}{NT}\sum_{i=1}^{N}\sum_{t=1}^{T}\left(y_{i,t}^{\text{sim}} - y_{i,t}^{\text{gt}}\right)^2 \tag{6.6}$$

where N is the number of effluent components (4 in this case), and T is the number of time steps. The goal of the optimization is to maximize this objective function, thereby finding a parameter configuration that produces effluent behavior similar to the ground truth.

6.2.5 Clustered Firefly Algorithm Experiment

The cFA is used to estimate the parameters to match the ground truth data. Its parameters were chosen based on prior experiments to ensure a reasonable runtime so that the algorithm gets enough time to explore the solution space but at the same time not too much

so that it becomes unpractical for real life scenarios.
Table 6.7 lists the chosen parameters for this experiment.

Parameter	**Cluster Firefly**
Initial number of fireflies (n_{max})	35
Number of generations	50
Minimum number of fireflies (n_{min})	5
Target Fireflies per cluster (n_{target})	5
Minimum fireflies per cluster	1
α (randomness weight)	0.5, decay factor 0.97
β_0 (initial attractiveness)	1.0
γ (light absorption)	Range 0.01 $\rightarrow$ 10

Table 6.7: Cluster Firefly Algorithm configuration used for ASM1 parameter estimation.

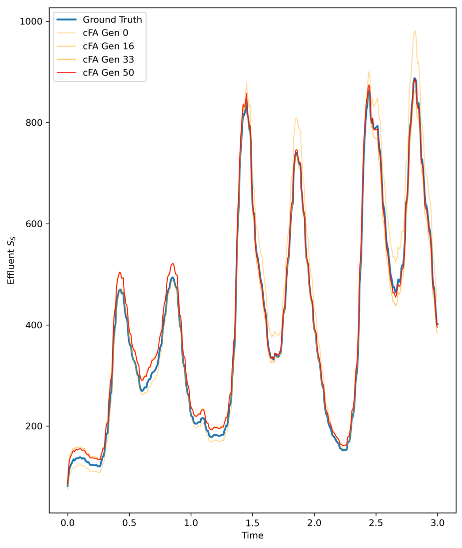
Results

It took 1 hour and 57 minutes to run the first experiment with the setup and the hardware that was described in Section 5.3.1.

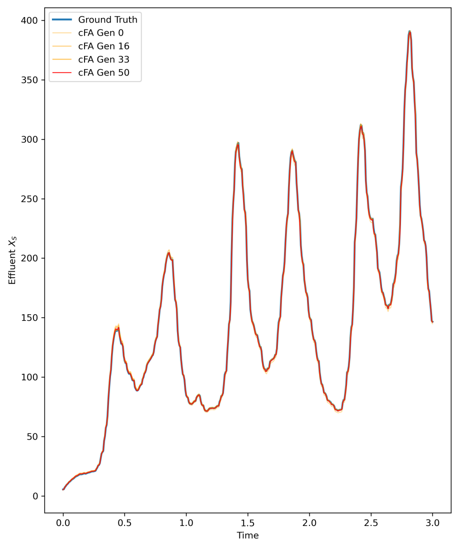
Figure 6.2 shows the simulated effluent concentrations of the 4 components compared to the ground truth curves over multiple generations of the clustered Firefly Algorithm. As expected, the simulated curves improve in later generations, gradually aligning more closely with the ground truth. In the final generation, all curves exhibit a close match.

However, it can be observed that the initial discrepancy between simulation and ground truth varies across components. For instance, early predictions of S_{NO} show a larger deviation compared to S_{NH}, which is already relatively close from the beginning. This suggests that some components are more sensitive to parameter changes or more difficult to fit accurately.

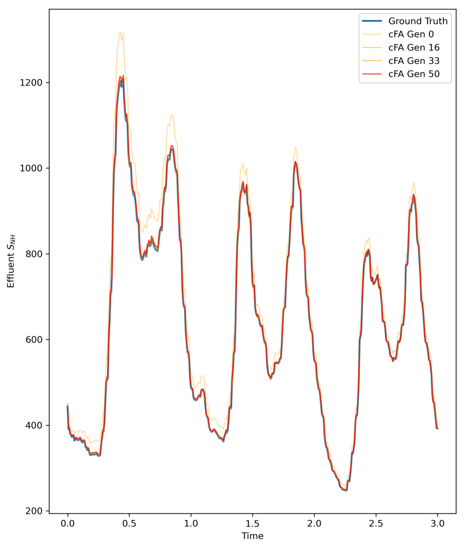
Despite differences in absolute values, all four simulated effluent curves exhibit the same qualitative behavior as the ground truth, including the position of peaks and the overall trend.



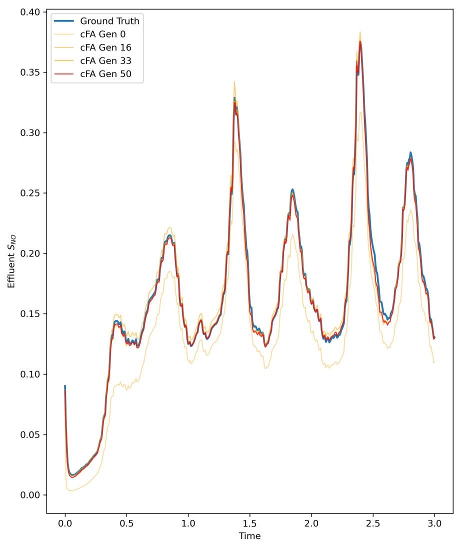
(a) Effluent S_S



(b) Effluent X_S



(c) Effluent S_{NH}



(d) Effluent S_{NO}

Figure 6.2: Simulated vs. ground truth effluent concentrations for the four monitored components over generations of the Clustered Firefly Algorithm.

Table 6.8 presents the ground truth values of the kinetic parameters alongside the estimates produced by the algorithm. While the estimated values do not exactly match the ground truth, most fall within a reasonably comparable range. Notably, the estimates for K_{NH} and K_S deviate more significantly—being approximately half and double their true values, respectively. Despite these discrepancies, the resulting curves closely match the ground truth curve.

	Y_H	b_H	μ_H	μ_A	K_{OA}	K_{NH}	K_S
Ground Truth Parameters	0.69	3.34	6.16	0.51	1.37	2.79	38.52
Estimated Parameters	0.67	3.34	8.46	0.49	1.30	1.28	68.06

Table 6.8: Ground Truth Parameters and Estimated Parameters

6.2.6 Generalization Performance

Since two out of seven estimated parameters—K_{NH} and K_S—deviate notably from the ground truth, yet the model still achieves a very good fit across all effluent values, it is important to investigate whether this performance generalizes to unseen data. This step helps determine whether the accurate fit results from overcompensation between parameters, which could indicate overfitting and limit the model's predictive reliability beyond the calibration timeframe.

This consideration is particularly important in the context of MPC, where the primary objective is not to replicate historical behavior, but to accurately predict future system states for effective control.

Therefore, the estimated parameters obtained from the initial 3-day calibration period were applied to simulate a 2-week timeframe. In this setup, the first 3 days correspond to the training data used for parameter estimation, while the remaining 11 days represent previously unseen data. By comparing the model's output with the ground truth simulation over the full period, the quality and generalization capability of the estimated parameters can be more thoroughly evaluated.

Results

Figure 6.3 shows the resulting effluent curves produced using the ground truth parameter values compared to those generated with the estimated parameters. The simulated curves match the ground truth very closely for all compounds, both during the estimation period and throughout the unseen timeframe.

Minor deviations are observed for component S_{NO} between days 6 and 8. However, these differences are small in magnitude, and the overall alignment between the two curves remains strong.

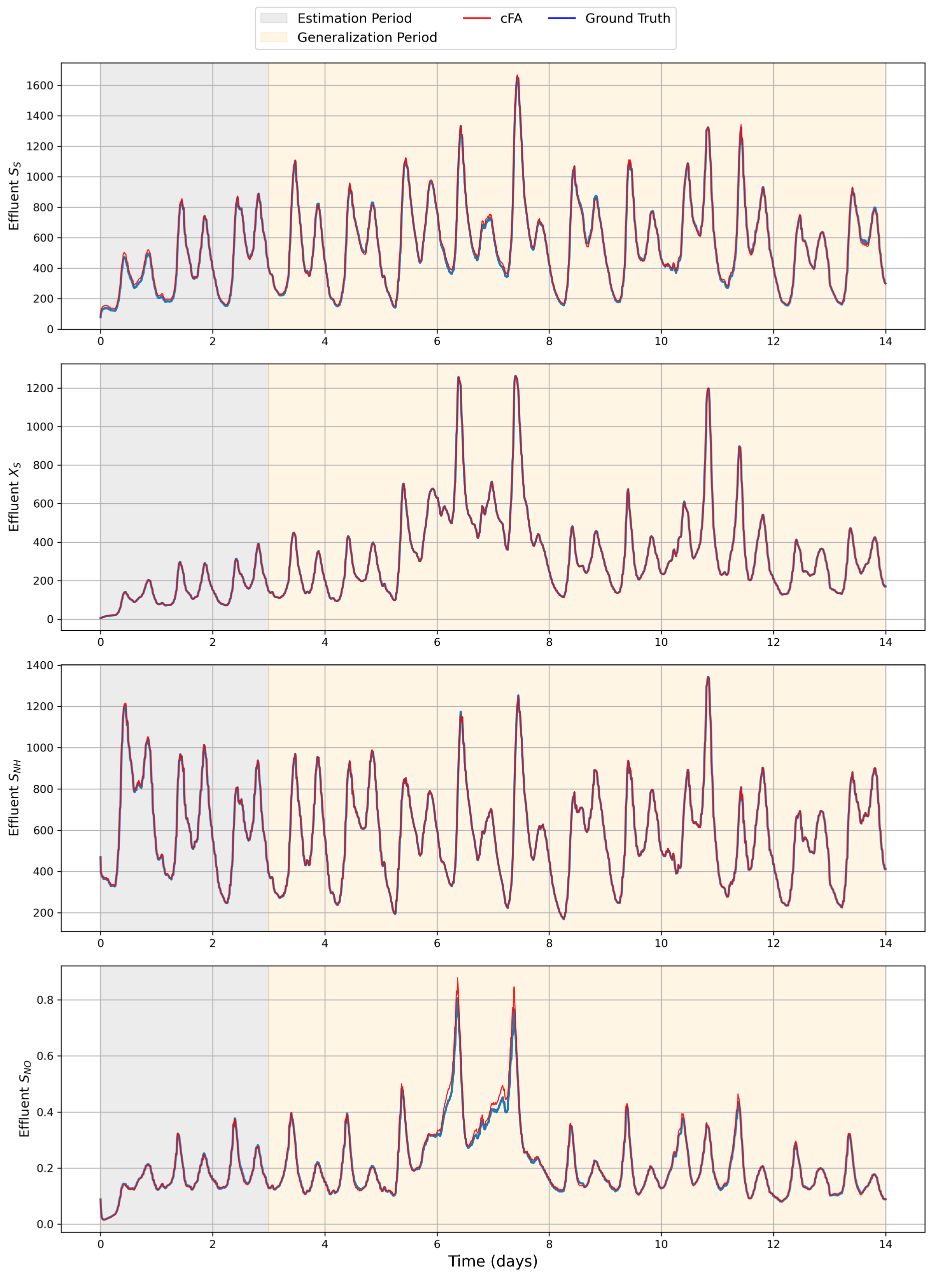


Figure 6.3: Comparison of simulated effluent concentrations using estimated parameters versus ground truth values across all components. The training period is highlighted in grey, and the test period in yellow.

6.3 Discussion

In this section, the findings of the parameter estimation on the synthetic effluent data are examined. While the algorithm achieved a good fit across all effluent components within a reasonable timeframe, several experimental limitations affecting the realism and practical applicability of the results are also highlighted.

Algorithmic Performance

The parameter estimation results for the simulated WWTP demonstrate that the cFA can reproduce the dynamic behavior of effluent components under idealized conditions. Although two of the seven estimated parameters deviated significantly from the ground truth, the resulting output curves still closely matched the simulated data. This indicates that accurate model fits can be achieved even when some estimated parameters differ substantially from their true biological values, suggesting a potential loss of interpretability. Nevertheless, these results highlight the algorithm's capability to explore and identify solutions within complex, non-linear optimization landscapes such as those defined by the ASM1 model.

This conclusion is further supported by the generalization experiment, where simulations using the estimated parameters continued to closely align with the ground truth curves beyond the training period. This consistency suggests that, in this experiment, the algorithm did not merely find a solution overfitted to the training data, but rather identified parameter sets that generalize well. Such behavior is especially valuable in predictive modeling and control, where the focus lies on accurate state prediction under previously unseen conditions.

Experimental Limitations

Even though the algorithm performed well in this setup, several important limitations should be noted. First, the simulation assumes idealized conditions: no measurement noise, perfect knowledge of the system's internal state, and continuous inflow data. These assumptions simplify the estimation task considerably, since the ground truth can be exactly generated by the same model structure. In contrast, real-world data is typically noisy, incomplete, and collected at lower sampling frequencies, which would introduce an inherent mismatch between model and measurements.

Second, the similarity in qualitative behavior between the simulated and ground truth effluent curves — even in early generations — is largely due to structural properties of the model. Because the outflow is strongly correlated with the inflow, and the same inflow pattern is used in both the ground truth and candidate simulations, some level of curve alignment is to be expected. This should not be attributed to the algorithm's performance, but rather to the model's deterministic structure.

Furthermore, standard values were assumed for parameters not included in the estimation process. While this is a reasonable simplification for a controlled test case, it is an unrealistic assumption in real applications, where insensitive parameters may vary from plant to plant as well.

Lastly, in the context of the generalization experiments, it is important to consider that in real-world systems, the biological composition of microbial communities in a WWTP may change over time. As a result, the kinetic parameters themselves could drift, leading to gradual changes in the system's dynamics. This kind of parameter drift was not modeled in the generalization experiment, and realistically simulating such changes in synthetic data is challenging. Consequently, the current setup does not allow for a meaningful assessment of the algorithm's robustness under conditions of evolving system dynamics, which are likely to occur in practical applications.

In conclusion, this experiment demonstrates that the cFA can handle the type of non-convex optimization problems posed by mechanistic WWTP models. However, its practical applicability cannot be judged based on this idealized scenario alone. Future experiments would need to incorporate realistic conditions such as partial observability, measurement noise, and possibly real-world data to more accurately assess the algorithm's utility in operational settings.

Chapter 7

Thermal Parameter Estimation from Heat Pump Data

Accurate thermal parameter estimation is a prerequisite for optimized control of heating systems using dynamic simulation models of real buildings. In a recent study, *Hasrat et al.* [9] demonstrated that Model Predictive Control (MPC), when combined with adaptive model identification, weather forecasts, and day-ahead electricity prices, can significantly reduce energy consumption and costs in residential buildings equipped with heat pumps. By employing a simplified, data-driven thermal model and synthesizing control strategies using Uppaal Stratego, their approach achieved energy cost reductions of up to 49%, depending on the control strategy used.

We adopt this use case for two main reasons. The first is to evaluate the effectiveness of the cFA in identifying suitable parameters from real-world data. The second is to compare its performance to already established methods for parameter estimation, particularly CTSM-R [31] in this context.

In this case study, we estimate thermal parameters—specifically, heat transfer coefficients—of a residential building based on real-world operational data provided by the company **CEDAR**[1]. The dataset includes indoor temperature measurements, outdoor weather conditions, and detailed heat pump operation data.

This experiment enables us to assess the robustness of the cFA under realistic conditions, including noisy measurements, missing data, and a simplified thermal model that abstracts away some of the building's physical complexities.

This use case provides a valuable addition for evaluating the performance of the cFA, as—unlike the previous case—it is based on real-world data from a physical system.

[1]CEDAR: `https://cedar-heat.dk/`

7.1 Thermal Model Description

The thermal model used in this experiment is directly taken from *Hasrat et al.* [9]. In their work, a building with four rooms is modeled, with each room having its own set of thermal parameters. This modeling approach can be easily extended to buildings of arbitrary size, as the parameters are estimated individually for each room.

In the model, the thermal dynamics of each room i are represented by a three-state model, consisting of the room temperature $\tilde{T}_r^i$, heater temperature $\tilde{T}_h^i$, and envelope temperature $\tilde{T}_e^i$. The evolution of these state variables is governed by state and external input variables (summarized in Table 7.1), heat exchange coefficients (provided in Table 7.2), and the system of differential equations given in Eqs. 7.1, 7.2, and 7.3.

Table 7.1: State and Input Variables

Variable	Description
$\tilde{T}_r^i$	room i air temp.
$\tilde{T}_h^i$	room i floor temp.
$\tilde{T}_e^i$	room i envelope temp
$\overline{M}^i$	room i water mass flow
$\overline{T}_{forward}$	water temp. exiting heat-pump
$T_{outdoor}$	outside temp
S^i	solar heat to room i

Table 7.2: Heat Exchange Coefficients

Coefficient	Description
α_h^i	heat resistance between floor and room air
α_e^i	heat resistance between envelope and room air
α_s^i	heat resistance between solar radiation and room
α_ω^i	heat resistance between water pipes and floor
α_a^i	heat resistnace between envelope and outdoor
α_n^i	heat resistance between room i and n
β_h^i	heat capacity of floor pipes
β_h^e	heat capacity of evelop

$$\frac{d\tilde{T}_r^i}{dt} = \alpha_h^i(\tilde{T}_h^i - \tilde{T}_r^i) + \alpha_e^i(\tilde{T}_e^i - \tilde{T}_r^i) + \alpha_s^i \cdot S^i \tag{7.1}$$

$$\frac{d\tilde{T}_H^i}{dt} = \frac{\alpha_h^i}{\beta_h^i}(\tilde{T}_r^i - \tilde{T}_h^i) + \alpha_\omega^i \cdot \overline{M^i}(\overline{T}_{forward} - \tilde{T}_h^i) \tag{7.2}$$

$$\frac{d\tilde{T}_e^i}{dt} = \frac{\alpha_e^i}{\beta_e^i}(\tilde{T}_r^i - \tilde{T}_e^i) + \alpha_a^i(T_{outdoor} - \tilde{T}_e^i) + \sum_{n \in N} \alpha_n^i(\tilde{T}_r^n - \tilde{T}_e^i) \tag{7.3}$$

Figure 7.1: Grouped representation of model components: input variables, heat exchange coefficients, and differential equations. The figure is taken from [9]. N in Equation 7.3 is the set of neighboring rooms for room i.

Since the model treats each room independently, the parameter estimation process must be performed separately for each room. This approach has the advantage that the dimensionality of each estimation problem remains comparatively small.

Variable	Description
$\tilde{T}_r^i$	Room i air temperature
$\overline{T}_{\text{forward}}$	Water temperature exiting the heat pump
T_{outdoor}	Outside temperature

Table 7.3: Variables directly available from the dataset.

7.2 Data

As previously mentioned, this experiment uses real-world data to estimate the heat exchange coefficients. All data is provided in the form of time series, where each entry consists of a measured value and its corresponding timestamp.
The data was collected during the winter months in northern Denmark. However, the exact timeframe has been intentionally omitted to protect privacy.

However, not all variables required by the model are directly available in the dataset. Some values have been estimated based on other data or derived using simplified model assumptions.

The measured data available for the heating model is summarized in Table 7.3. For the missing variables, different estimation approaches have been applied.

It should also be noted that the dataset used in this thesis did not include the exact number of rooms in the building, as a single room may contain multiple temperature sensors. Additionally, no information about room adjacency was provided. As a result, both the room count and neighborhood structure had to be assumed, as will be described in Section 7.3.

7.2.1 Estimating the Water Mass Flow $\overline{M^i}$

The company **CEDAR**, which provided the data for this experiment, mentioned that while "supply pump speed data" could serve as a proxy for water mass flow, it is considered unreliable in practice. Instead, they recommended the following approach (translated from danish):

> It is possible to abstract away from the mass flow component of the equations and assume that the mass flow decreases from 100% to 0% over a narrow interval, as the average room temperature increases from approximately 21.5°C to 22.5°C.

Following this approach, we define the mass flow for room i as a function of the room temperature $\tilde{T}_R^i$ using a sigmoid function, so its output lies in the interval $[0, 1]$:

$$\overline{M^i}(\tilde{T}_R^i) = \frac{1}{1 + \exp\left(k(\tilde{T}_R^i - T_{\text{set}})\right)}, \tag{7.4}$$

where:

- T_{set} is the setpoint temperature that determines the center of the transition region.
- k controls the steepness of the transition.

Both T_{set} and k are treated as additional parameters to be optimized as part of the room model.

7.2.2 Estimating Solar Heat S^i to Room i

Direct measurements of solar heat were not available in the dataset. However, the weather data included two relevant variables: the *cloud area fraction*, indicating the proportion of the sky covered by clouds, and the *sun altitude*, corresponding to the solar elevation angle γ.

Following the work of *Kasten and Czeplak* [40], the solar irradiance under clear-sky conditions can be approximated as proportional to $\sin(\gamma)$. Since the room-specific solar gain is scaled by the heat transfer coefficient α_s^i, it is sufficient to estimate the *relative* solar gain; the absolute scale can be absorbed into α_s^i during parameter fitting.

To account for cloud cover, a nonlinear attenuation factor is applied based on the cloud area fraction $c \in [0, 1]$:

$$S^i = \sin(\gamma) \cdot (1 - 0.75 \cdot c^{0.75}) \tag{7.5}$$

This formulation models the reduction in solar gain due to cloudiness more realistically than a simple linear scaling and is empirically supported for mid-latitude atmospheric conditions [40].

Because the dataset does not provide room-specific solar exposure (e.g., orientation or shading), S^i is assumed to be the same for all rooms. This simplification can be compensated by the solar heat coefficient α_s^i, which captures differences in actual solar impact across rooms.

7.2.3 Start values for state variables

From Table 7.3, it is apparent that the floor temperature $\tilde{T}_h^i$ and the envelope temperature $\tilde{T}_e^i$ are not available in the dataset. Therefore, their initial values are treated as parameters to be optimized as well.

7.2.4 Interpolation of Time Series Data

The data originates from various sources—different sensor types for room temperature measurements, a heat pump sensor, and weather forecast data—which differ in both their start and end times, as well as in their sampling frequencies. To address this, a common time frame (start and end point) had been identified, and a dataset with one-minute intervals was generated using linear interpolation.

Most sensor data was originally recorded every 30 minutes. However, since temperature tends to change slowly, linear interpolation was deemed sufficiently accurate.

7.3 Identifying the Room Layout

As described in Section 7.1, the parameters are estimated for each room individually. The model accounts for heat exchange between neighboring rooms, as shown in Equation 7.3, which describes the evolution of the envelope temperature $\tilde{T}_e^i$ for room i.

However, the dataset does not include any information about the physical layout of the rooms. As a result, the adjacency between rooms is unknown.

To enable the use of the model despite this limitation, the room layout is inferred and represented as an undirected graph based on correlation analysis.

7.3.1 Correlation Analysis

The interpolated temperature data from the rooms and the outside temperature over a two-week period is used for the correlation analysis.

Figure 7.2a shows the correlation matrix from the initial analysis. It is noticeable that some rooms exhibit very high correlations (1.0 or close to it). These rooms are likely to represent the same physical space, as one room may contain multiple temperature sensors, and two different sensor types are used for room measurements.

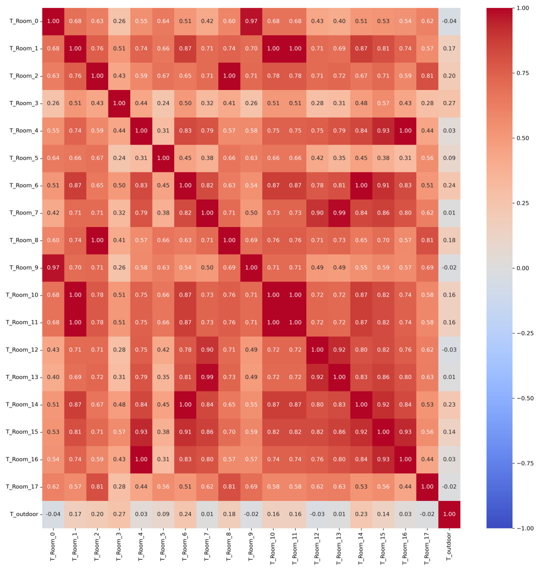
Since including duplicates in the parameter analysis would not be meaningful, only one sensor per highly correlated room group was retained. A cutoff value of 0.96 was arbitrarily chosen—rooms with a correlation equal to or higher than this threshold were considered to be the same room.

After the assumed duplicates were removed, a total of 11 rooms remained. A second correlation analysis was performed with the remaining rooms. The result is shown in Figure 7.2b. Here, it can be noted that the number of rooms for parameter estimation was cut down from 17 to 11.

7.3.2 Room Layout Graph

Using the second correlation matrix, an undirected graph was constructed to serve as the assumed room layout. Two rooms are considered neighbors if their correlation exceeds a threshold, which was arbitrarily set to 0.5.

Figure 7.3 shows the resulting undirected graph. It is noticeable that most rooms exhibit a high level of interconnectivity, while others—such as Room 3 and Room 5—have only two and four neighboring rooms, respectively. Although the layout may not accurately reflect the true physical structure of the building, it provides a practical basis for applying the thermal room model as described earlier.


(a) Correlation Matrix before rooms were removed.

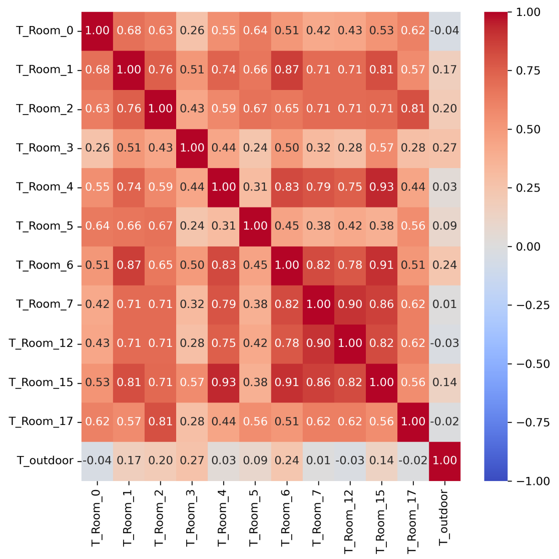

(b) Correlation matrix after rooms were removed.

Figure 7.2: Heatmaps for correlation analysis of room temperatures.

7.4 Experiments

Based on the previously described data, the assumed room layout, and the thermal room model, the parameters for each room were estimated using the cFA. A time span of one week was considered. The objective of the experiment was to identify parameters such that the simulated room temperatures over the 7-day period closely matched the observed temperature data.

In addition, the parameter estimation results obtained using the cFA were compared to those from CTSM-R—a tool used in the original publication [9] that introduced the thermal grey-box model.

Finally, the generalization performance of the estimated parameters was evaluated on a separate time period that was not used during training, in order to assess how well the model fit temperature data beyond the initial one-week window.

7.4.1 Clustered Firefly Parameter Estimation

To estimate the thermal parameters of each room, the cFA was employed.

In contrast to previous experiments, parallel computing was used to take advantage of all available CPU cores in order to reduce computation time. The same hardware that was already described in Section 5.3.1 was used.

Hyperparameters were selected based on prior experiments to strike a balance between runtime and solution quality. Table 7.4 lists the parameters used. The same settings were

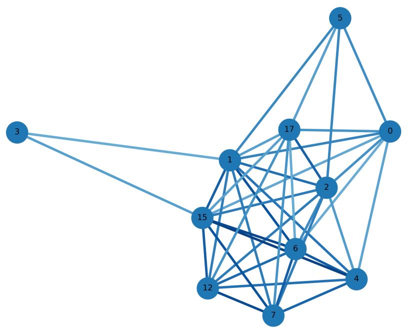


Figure 7.3: Assumed room layout for parameter estimation problem.

applied to all rooms.

Parameter	**Cluster Firefly**
Initial number of fireflies (n_{max})	70
Number of generations	70
Minimum number of fireflies (n_{min})	12
Fireflies per cluster	7
Minimum fireflies per cluster	1
α (randomness weight)	0.5, decay factor 0.97
β_0 (initial attractiveness)	1.0
γ (light absorption)	Range $0.01 \to 10$

Table 7.4: Clustered Firefly Algorithm configuration used for thermal parameter estimation.

Table 7.5 summarizes the parameter bounds used in this experiment. For the heat exchange coefficients, a wide range was selected, as no expert knowledge was available. These ranges are similar to those used in the experiments of *Hasrat et al.* [9].

Realistic bounds were applied to the initial values of the temperatures T_h^i and T_e^i. For the mass flow estimation, the setpoint temperature T_{set} was also assigned a realistic range, while a narrow, arbitrarily chosen range was used for the parameter k to control the slope of the mass flow function.

Objective Function

Similar to the WWTP use case, the objective function is based on comparing the simulated output time series for a given parameter configuration with the measured data. In this case, the simulated room temperature is compared to the observed room temperature.

Parameter	Description	Range
α_a^i, β_h^e, etc.	Heat Exchange Coefficients	[0.0001, 1000]
T_h^i, T_e^i	Initial temperatures (°C)	[10, 40]
k	Mass flow parameter	[0.1, 10]
T_{set}	Setpoint temperature (°C)	[19, 23]

Table 7.5: Parameter bounds used for cFA Parameter estimation

One objective function used is the negative mean squared error (MSE), as defined in Equation 6.6. Note that the MSE is not normalized in this context, since we are only comparing a single output variable—the estimated room temperature—to its measured counterpart. In the WWTP use case, normalization was used to balance the influence of multiple output time series, which is not necessary here.

In this thesis, we propose a **shape-weighted MSE (swMSE)** objective function to further encourage models that not only minimize overall error but also better capture system dynamics. In this function, the *Pearson correlation coefficient* [41] is used to scale the negative MSE, effectively penalizing output curves that do not exhibit similar dynamic behavior. The objective function is mathematically defined as follows:

$$\text{swMSE} = \frac{\text{MSE}(y^{\text{sim}}, y^{\text{obs}})}{\max\left(\delta + (1-\delta)r,\ 1\text{e}{-3}\right)} \tag{7.6}$$

where:

- MSE is the (negative) mean squared error,
- y^{sim} is the simulated temperature time series,
- y^{obs} is the observed temperature time series,
- $\delta \in [0, 1]$ is a hyperparameter controlling the influence of the correlation factor, set to 0.5 in our case,
- r is the Pearson correlation coefficient between the two time series.

The `max` function in the denominator is used to handle cases where the correlation is negative, ensuring that the scaling remains stable and preventing unintended behavior in the objective function.

Results

Figure 7.4 presents graphs showing the measured temperature in each room, along with the simulated temperatures generated using the best parameter configuration identified by the cFA. Results are shown for both the swMSE and the standard MSE objective functions.

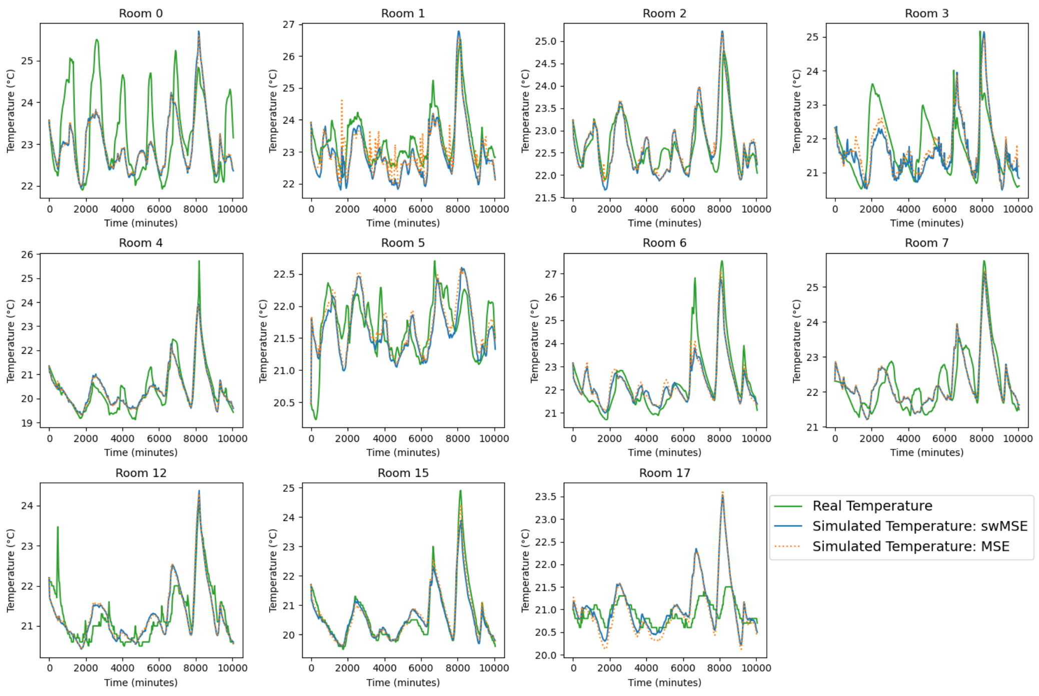


Figure 7.4: Comparison of simulated and observed room temperatures over time for the clustered Firefly Algorithm.

Some of the graphs reveal significant discrepancies between the real and simulated temperature curves, both in terms of dynamics and absolute values. This suggests that the clustered Firefly Algorithm was not able to identify parameter sets that accurately capture the thermal behavior of all rooms.

However, in most cases (especially rooms 2, 4, and 15), the simulated temperatures align very closely with the measured data. Although not perfect, these simulations approximate the real temperature evolution more accurately than in other rooms.

Most of the results obtained using the two different objective functions are closely aligned, with the exception of room 1, where both simulations deviate noticeably from the actual temperature trajectory.

Table 7.6 presents the objective function values and computation times for each room using the clustered Firefly Algorithm, run with both the swMSE and standard MSE objective functions. As the objective function is being maximized, higher values (i.e., values closer to zero) indicate better performance. Generally, rooms with better alignment between simulated and observed temperatures also exhibit higher objective values.

Note, however, that the objective values obtained using the two functions are not directly comparable, as they reward different aspects of model fit.

It is also notable that, despite using the same algorithmic configuration for each room, the computation time varies considerably across rooms. The computation times for the two different objective functions are generally similar; in some cases, the swMSE objective results in shorter runtimes, while in others, the MSE objective is faster.

Room	**swMSE**		**MSE**	
	Value	**Time [s]**	**Value**	**Time [s]**
0	-0.855	1194.08	-0.643	1642.41
1	-0.312	834.81	-0.325	1255.26
2	-0.200	1185.03	-0.175	1140.62
3	-0.638	3914.35	-0.488	1447.00
4	-0.130	11968.06	-0.121	12609.74
5	-0.184	1543.99	-0.148	1155.00
6	-0.442	967.89	-0.403	1090.18
7	-0.217	1162.31	-0.219	953.72
12	-0.150	1450.87	-0.151	1907.85
15	-0.088	12795.10	-0.052	13113.53
17	-0.338	1432.08	-0.300	1596.71

Table 7.6: Objective function values and computation times for the swMSE and MSE objective functions using the clustered Firefly Algorithm.

The specific parameter values that resulted from the cFA are provided in Appendix B.1 and B.2.

7.5 Parameter Estimation using CTSM-R

To provide a benchmark for the parameter estimation results obtained using the cFA, we also performed parameter estimation using the tool CTSM-R.

CTSM-R (Continuous Time Stochastic Modeling in R) is a modeling and estimation framework for continuous-time grey-box systems with stochastic components [31]. It uses maximum likelihood estimation and Kalman filtering to identify parameters for a system of stochastic differential equations based on observed time series data. CTSM-R was also used in the original study that introduced the interconnected room temperature model by *Hasrat et al.* [9], making it a natural choice for comparison in this experiment.

In CTSM-R, the system is described by a continuous-discrete time state space model, consisting of stochastic differential equations for the system dynamics and discrete-time measurement equations as defined in [31]:

$$d\mathbf{x}_t = f(\mathbf{x}_t, \mathbf{u}_t, t, \boldsymbol{\theta})\, dt + \sigma(\mathbf{u}_t, t, \boldsymbol{\theta})\, d\boldsymbol{\omega}_t \tag{7.7}$$

$$\mathbf{y}_k = h(\mathbf{x}_k, \mathbf{u}_k, t_k, \boldsymbol{\theta}) + \mathbf{e}_k, \quad \mathbf{e}_k \sim \mathcal{N}(0, \mathbf{S}(\mathbf{u}_k, t_k)) \tag{7.8}$$

Here:

- $\mathbf{x}_t$ denotes the (unobserved) state vector at time t,
- $\mathbf{u}_t$ is the exogenous input vector,
- $\boldsymbol{\theta}$ represents the set of model parameters,
- $f(\cdot)$ and $\sigma(\cdot)$ define the drift and diffusion terms of the system,
- $d\boldsymbol{\omega}_t$ is a Brownian motion process,
- $\mathbf{y}_k$ are the measurements at discrete time points t_k,
- $h(\cdot)$ is the measurement function,
- $\mathbf{e}_k$ is the Gaussian measurement noise with covariance $\mathbf{S}(\mathbf{u}_k, t_k)$.

Setup

The setup of this experiment closely follows the procedure used in [9].

To apply the three-state energy model described in Section 7.1, each differential equation is extended with a diffusion term $\sigma(\cdot)$, as defined in Equation 7.7. This term introduces stochasticity into the system through process noise.

For each state equation i, the diffusion term is defined as:

$$\exp(p_i) \cdot dw_t,$$

Parameter	Description	Initial Value	Range
α_a^i, β_h^e, etc.	Heat Exchange Coefficients	0.2	[0.0001, 1000]
T_h^i, T_e^i	Initial temperatures (°C)	T_r^i	[10, 40]
k	Mass flow parameter	5	[0.1, 10]
T_{set}	Setpoint temperature (°C)	21	[19, 23]
p_i	Process Noise	0.01	[-100,100]
e_{11}	Measurement Noise	0	[-100, 100]

Table 7.7: Initial parameter values for CTSM-R experiment.

where p_i is a parameter estimated during model fitting. This parameter determines the magnitude of the process noise affecting the dynamics of state i. This allows the model to capture small unmodeled disturbances and uncertainties inherent in real-world temperature dynamics.

The observation model links the simulated room temperature state T_r^i to the measured room temperature variable T_{room}. This is expressed in CTSM-R as:

$$T_{\text{r,obs}}^i = T_r^i + e_k, \quad e_k \sim \mathcal{N}(0, \exp(e_{11})^2)$$

Here, e_k denotes Gaussian measurement noise with zero mean and a variance of $\exp(e_{11})^2$, where e_{11} is a parameter estimated during model fitting. The exponential transformation ensures that the variance remains strictly positive throughout the optimization.

This formulation allows the model to account for measurement uncertainty in the observed temperature signal.

For both the process and measurement noise parameters, the lower and upper bounds were adopted directly from [9].

In addition, for the heat exchange coefficients and other physical model parameters, the same parameter ranges as used for the clustered Firefly Algorithm were applied (see Table 7.5).

In CTSM-R, initial values must be provided for each parameter, serving as starting points for the estimation process. Based on prior experiments, starting values close to the lower bounds tended to result in faster convergence. Therefore, small initial values were used in our experiments. An overview over the initial values and the parameter bounds for all parameters is given in Table 7.7.

Results

CTSM-R successfully estimated parameters for the majority of the rooms, allowing their temperature trajectories to be simulated. However, for rooms 0, 3, 4, and 7, the algorithm failed to converge to a good solution with error messages such as:

-
  ```
  Did not converge.  I'm sorry, but I don't know this error code.
  Code:  130
  ```

- `Did not converge. The state covariance matrix is not positive definite. Code: 30`

For these rooms no valid parameter set was obtained for simulation.

Figure 7.5 shows the simulated room temperatures obtained from the CTSM-R parameter estimation, alongside the real observed temperatures. For comparison, the corresponding simulation results using the cFA with the swMSE objective function are also included.

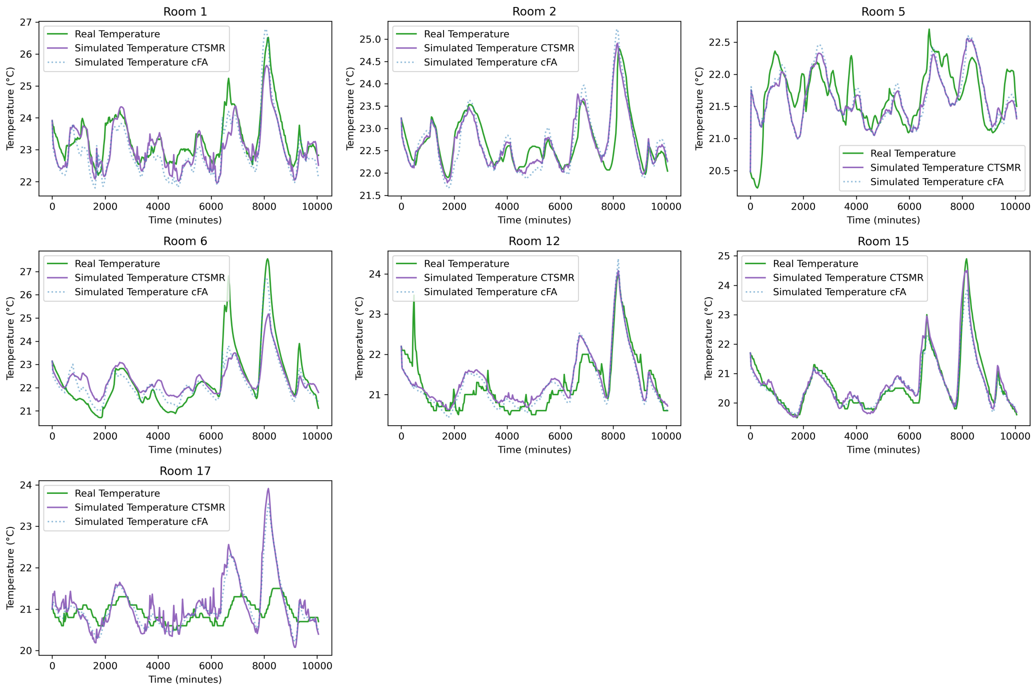


Figure 7.5: Simulated vs. observed room temperatures for CTSM-R and cFA across all rooms.

For most rooms where parameters were successfully identified, the simulated temperature curves align closely with those generated using the cFA-derived parameters. For rooms where the clustered Firefly Algorithm struggled to find a well-fitting parameter set (e.g., Rooms 5 and 17), CTSM-R also fails to produce good results. This suggests that the simplified model may not adequately capture the thermal dynamics of these rooms.

To examine the performance in more detail, Table 7.8 presents the negative MSE values and computation times for the CTSM-R results. For rooms where no solution was found, only the computation time is reported.

For comparison, the corresponding MSE values and runtimes from the cFA are also included. With the exception of Room 7, CTSM-R completes significantly faster than cFA across all rooms. In total, CTSM-R required 5,070.92 seconds to run for all rooms, while cFA took 39,648.57 seconds—approximately eight times longer.

In addition to faster execution, CTSM-R achieves similar or even better MSE values for most rooms.

Room	CTSM-R		cFA (swMSE objective)	
	MSE Value	Time [s]	MSE Value	Time [s]
0	-	125.7	-0.673	1194.08
1	-0.254	499.7	-0.301	834.81
2	-0.131	884.93	-0.170	1185.03
3	-	172.45	-0.525	3914.35
4	-	1053.49	-0.125	11968.06
5	-0.153	88.79	-0.152	1543.99
6	-0.691	47.8	-0.393	967.89
7	-	1943.62	-0.215	1162.31
12	-0.145	57.83	-0.144	1450.87
15	-0.086	152.36	-0.084	12795.10
17	-0.412	43.15	-0.274	1432.08

Table 7.8: MSE objective values and computation times for the results of CTSMR and clustered Firefly Algorithm using the swMSE objective function.

The complete set of parameter values estimated by CTSM-R is provided in Appendix B.3.

7.5.1 Generalization Performance

As described earlier, the goal of parameter estimation in this context is to obtain a model that accurately captures the thermal dynamics of the different rooms and can be used to predict future states (i.e., room temperatures) under varying control strategies. This, in turn, enables the derivation of optimal strategies that minimize cost and energy usage during heat pump operation.

Therefore, it is important that the estimated parameters generalize well beyond the training data. To assess this, we evaluate how the fitted parameters perform when the time window is extended. In this experiment, a two-week period was considered: the first week was used for parameter estimation, and the second week—comprising previously unseen data—was used to evaluate generalization performance.

Results

Figure 7.6 shows the temperature trajectories over the two-week period, including the observed temperatures, the simulated temperatures from the clustered Firefly Algorithm using the swMSE objective function, and the simulated temperatures from CTSM-R. For rooms where CTSM-R failed to produce a solution, only the Firefly-based simulation is shown.

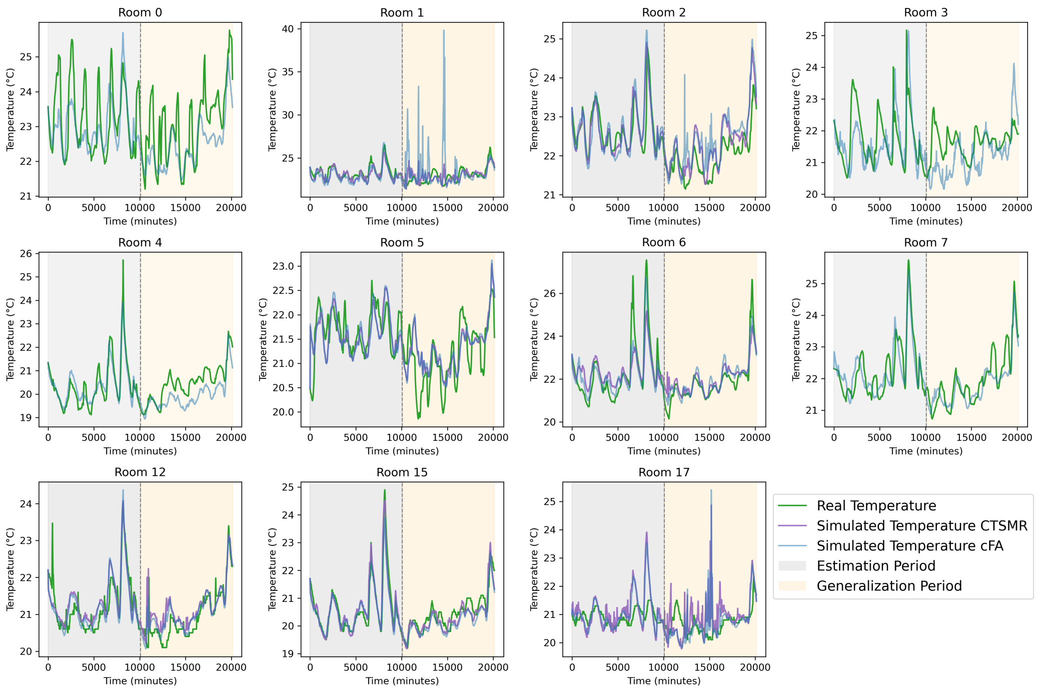


Figure 7.6: Observed and simulated room temperatures over a two-week period. Simulations were generated using the clustered Firefly Algorithm (cFA) with the swMSE objective function and CTSM-R. Only cFA results are shown for rooms where CTSM-R failed to converge. The first week was used for parameter estimation; the second week tests generalization.

The clustered Firefly Algorithm produced mixed results. For some rooms where a good temperature fit was achieved during the estimation period, the model also generalizes well to the second week—see, for example, rooms 7, 12, and 15. However, in other cases—such as rooms 1 and 2—where the fit during the first week was acceptable, the second week shows large temperature spikes that deviate significantly from the observed data. As expected, for rooms where cFA failed to produce a good fit during the estimation period, for example room 17, the simulation also performs poorly in the generalization period.

CTSM-R generally achieved better generalization performance in cases where it closely matched the observed temperature during the first week—for example, rooms 1, 12, and 15. The simulations based on CTSM-R parameters do not exhibit unrealistic temperature spikes, with the exception of room 17.

For room 17, CTSM-R performs comparably poorly to cFA. Naturally, performance cannot be evaluated for rooms where CTSM-R failed to produce a valid parameter set during the estimation period. However, this inability to produce a solution can itself be interpreted as an indicator of performance in those cases.

Table 7.9 compares the negative MSE values for the second week. For room 1, where the cFA solution exhibited the most pronounced and unrealistic temperature spikes, the error is by far the highest. CTSM-R achieved better MSE values in 4 out of the 7 cases. As expected, the MSE values are generally higher compared to those in the parameter estimation experiment, which used only the training week (see Table 7.8).

Room	CTSM-R	cFA (swMSE)
0	-	-0.748
1	-0.487	-5.228
2	-0.236	-0.388
3	-	-0.619
4	-	-0.395
5	-0.226	-0.231
6	-0.251	-0.183
7	-	-0.189
12	-0.158	-0.098
15	-0.088	-0.104
17	-0.539	-0.393

Table 7.9: Comparison of MSE values for CTSM-R and the clustered Firefly Algorithm (cFA) using the swMSE objective function of the second week.

7.6 Discussion

This section discusses the findings of the thermal parameter estimation experiments using the cFA on real-world data from a residential heat pump system. The goal was to evaluate the algorithm's ability to identify thermal parameters that would allow simulated temperatures to closely match measured real-life data.

While the algorithm was able to identify suitable parameters for some rooms, it failed to produce meaningful parameter sets for others. This highlights a number of limitations and considerations that may explain the observed results—ranging from model simplifications and data quality issues to aspects of the experimental setup itself. The following discussion evaluates these factors and considers their implications for practical deployment and future research.

7.6.1 Model and Data Limitations

Model Simplification and Data Availability The use of a simplified thermal model, combined with missing or estimated data, likely contributed to the discrepancies observed between the simulated and measured room temperatures.

The three-state thermal room model itself is a simplification, as it does not account for various internal activities within the building. Examples include cooking, showering, opening windows for ventilation, or the presence of occupants—all of which can significantly influence room temperature but are not captured by the model. As a result, an inherent mismatch arises between the simulated temperatures and the actual conditions.

In addition, several assumptions had to be made to apply the model, which may have negatively affected performance. The model for room i requires knowledge of adjacent room temperatures to simulate inter-room heat exchange. Since this information was not available, the room layout was approximated using correlation analysis, from which an undirected graph of room connectivity was derived. As described in Section 7.3, rooms with very high mutual correlation were assumed to represent the same physical space, and neighboring rooms were selected based on an arbitrarily chosen threshold value. This process may have led to a room layout that does not accurately reflect the true physical structure of the building.

More sophisticated methods for inferring room adjacency—or additional experiments to determine more robust threshold values—could potentially improve the accuracy of the simulation. Alternatively, obtaining the actual adjacency information from building plans or available sensor metadata would likely be more reliable and is often obtainable in real-world settings—if such data is accessible.

Approximation of Mass Flow and Solar Gain Some critical data affecting the temperature trajectory in each room—such as mass flow and solar gain—had to be approximated, introducing an additional layer of simplification.

Mass flow was approximated using a simple heuristic: it was assumed to increase within a narrow range around a temperature setpoint. While this approach may provide a rough approximation, it does not reflect the actual control behavior of the heating system and may therefore have negatively affected the model's accuracy. A more refined approach for approximating mass flow could potentially improve model performance. The company CEDAR, which provided the dataset, also noted an alternative (translated from danish):

> It is also possible to infer it from rooms that have a "state" field indicating whether their flow valve is open.

This method, which infers mass flow based on the open/closed state of room valves, was not pursued due to time constraints but may offer a more accurate approximation.

Similarly, solar gain was not derived using a detailed radiation model, but rather assumed uniformly across all rooms. It was approximated based on available weather data—specifically solar altitude and cloud cover. While this offers a general sense of solar input, using more granular data (such as the fraction of low-, mid-, and high-altitude clouds) could improve the fidelity of the approximation. Moreover, the current approach does not differentiate between rooms, assuming the same solar exposure for all. Including room orientation data (which was not available) could enable more accurate, room-specific solar gain profiles and further enhance model realism.

7.6.2 General Performance, Comparison with CTSM-R, and Generalization

The experiments comparing the cFA with the CTSM-R tool revealed both advantages and disadvantages for each method. For the rooms where CTSM-R successfully converged to a solution, it achieved significantly shorter runtimes than cFA, while yielding parameter sets with comparable performance. This is likely due to CTSM-R employing local optimization techniques, which search for solutions close to the initial parameter estimates.

In contrast, the cFA performs global optimization, which naturally requires more computation time to explore a broader search space. However, this global nature brings the advantage of independence from initial parameter configurations. As a result, cFA was able to identify reasonable parameter sets for several rooms where CTSM-R failed to converge. This indicates that although CTSM-R can be faster when it succeeds, it is sensitive to initial conditions—potentially limiting its robustness in complex or noisy estimation tasks where good initial guesses are not available.

One potential enhancement for CTSM-R would be to run it from multiple initial guesses and select the best resulting solution. However, this raises questions about whether its runtime advantage would still hold, and how to systematically choose those initial guesses.

In the context of MPC with parameter estimation being an integral part of the control loop, it is crucial that the algorithm reliably returns a solution. Failure to do so may require fallback strategies, such as defaulting to pre-tuned parameters or previously identified configurations. In this regard, the cFA demonstrates greater robustness, as it always returns a solution. Of course, the quality of that solution is not always guaranteed to be optimal—as seen in the subpar fits for Room 1 and Room 17—but it avoids outright failure.

The runtime of cFA could potentially be improved by reducing the learning budget (e.g., number of fireflies or generations), which was set relatively high in these experiments. In many cases, the best solution was found in early generations. A promising enhancement would be to introduce a convergence criterion that stops the algorithm early if no significant improvement is observed over time. This would allow the algorithm to use its full budget for difficult estimation problems, while terminating early in easier cases—thereby improving overall efficiency without sacrificing robustness.

Generalization The generalization experiments revealed some limitations of the cFA. In some cases—most notably in Room 1—the algorithm achieved a good fit during the training week but produced unrealistic temperature spikes when applied to unseen data in the second week. This behavior indicates that the model was overfitted to the training data.

Such overfitting is problematic in practical applications where historical data is used to identify model parameters and the resulting model is expected to predict future system states for optimal control. In contrast, CTSM-R demonstrated better generalization performance for the rooms where it successfully converged, maintaining stable and realistic behavior on unseen data.

One possible reason for the unrealistic behavior observed in some rooms is the use of extremely wide—and in some cases, physically implausible—parameter bounds for the heat transfer coefficients. The bounds for these coefficients ranged from 0.0001 to 1000,

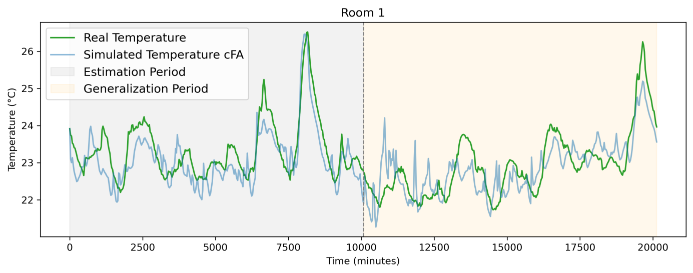


Figure 7.7: Simulated temperature vs observed temperature for room 1 with restricted parameter bounds.

based on those used in the original paper introducing the three-state model [9].

However, when examining the parameter values identified by CTSM-R, it becomes clear that such extreme values are rarely explored. For instance, the highest value found by CTSM-R across all rooms was 12.42 for the coefficient a_1^2 in Room 2. This suggests that the bounds were chosen to avoid unnecessarily restricting the search space, rather than to reflect physically realistic expectations.

In contrast, the clustered Firefly Algorithm did produce values close to the upper bound, such as 992.4 for coefficient a_4^{15} in Room 15. A possible consequence of this is that the algorithm may select physically unrealistic parameter combinations that nonetheless achieve a good fit during the training week by compensating for each other. However, on unseen data—where these compensatory dynamics may not hold—such parameter sets can lead to unrealistic or unstable behavior.

These findings suggest that while cFA is more robust in consistently producing solutions, its use in predictive control applications may require safeguards against overfitting—such as tighter parameter bounds or regularization.

To investigate this further, the parameter estimation process for Room 1 was repeated with the upper bounds for all thermal parameters restricted to 15.

As shown in Figure 7.7, this modification resulted in a significantly better fit: the temperature spikes previously observed were no longer present, suggesting that the generalization issues may indeed stem from unrealistic parameter bounds.

Future Work Future experiments could explore the impact of using narrower, more physically informed parameter bounds for the heat transfer coefficients of all rooms. This could reduce the risk of overfitting and improve the physical plausibility of the identified models. Additionally, introducing convergence criteria for the cFA—such as early stopping when no significant improvement is observed—may reduce runtime while preserving solution quality. These adjustments would be particularly relevant in real-time applications, such as model predictive control, where both performance and computational efficiency are critical.

Another potential direction would be to explore a hybrid approach, where the cFA is run for a limited number of generations to identify promising regions in the parameter space. These candidate solutions could then be used as initial values for CTSM-R, enabling fast and focused local optimization. Such a combination could leverage the global exploration capabilities of cFA and the efficient convergence of CTSM-R.

Chapter 8

General Discussion

This chapter discusses the overall results, design choices, and potential limitations of the *Clustered Firefly Algorithm*. It also outlines areas for improvement and possible directions for future development.

General Applicability

A general advantage of most nature-inspired global optimization algorithms—including the cFA—is their broad applicability to a wide range of problems and systems. These methods do not require gradient or Jacobian information, making them suitable even when the objective function is non-differentiable, noisy, or only accessible through simulations. As long as a simulation tool exists for a given system, the algorithm can be applied directly, making it particularly attractive for complex models where traditional gradient-based methods are impractical.

To give an example, an attempt was made to use CTSM-R for the WWTP use case, similar to its application in the residential building model. However, this revealed several potential shortcomings of CTSM-R in handling complex systems such as WWTPs.

CTSM-R is primarily designed for continuous-time systems, whereas WWTPs are typically modeled as hybrid systems. Discontinuities arise due to changing control setpoints, variable inter-tank water flows, and internal model components such as settling tanks, which use `min` functions and if-else logic. These features introduce non-smooth or discontinuous dynamics that challenge gradient-based estimation tools.

An effort was made to approximate the WWTP model as a continuous system to make it compatible with CTSM-R. This process involved re-implementing the model equations in R and modifying several model components to enforce continuity. Despite the significant additional workload, CTSM-R ultimately failed to produce parameter estimates for the continuous approximation of the WWTP model. The estimation process terminated with error messages such as:

- `90` – Unable to perform numerical ODE solution.
- `30` – The state covariance matrix is not positive definite.

These errors indicate numerical instability during simulation and filtering, likely caused by discontinuities or ill-conditioned dynamics, despite efforts to enforce continuity.

It should be noted that this is not to say CTSM-R is inherently unsuitable for complex models such as WWTPs. The issues encountered may have been due to how the continuous approximation of the model was implemented, rather than limitations of the tool itself. However, this experience highlights a clear practical advantage of the cFA: it can be applied directly to a wide variety of complex systems with minimal additional effort. In contrast, other estimation tools may require careful and sometimes non-trivial modifications to the model or its structure before they can be effectively used.

Comparison to Other Algorithms

In the benchmark experiments, the cFA was shown to outperform the original Firefly Algorithm, the modified FA variant, and Grid Search on the majority of the considered benchmark problems. For the WWTP case study, only the cFA was applied, while in the residential building case study it was compared to CTSM-R. In this comparison, cFA generally achieved a good fit and demonstrated more robust convergence behavior, although some generalization issues were observed, as discussed in Section 7.6.2.

While the results across all three experiments and use cases are promising, a more comprehensive evaluation would require comparisons with additional optimization algorithms. In particular, there exists a wide range of other global, nature-inspired optimization methods that could have served as alternative baselines. Including such comparisons would provide a stronger basis for assessing whether the cFA offers truly superior performance. However, due to time constraints, these additional experiments were not conducted in the current work.

Modifications to the Firefly Algorithm

In addition to the clustering mechanism and the division into local and global firefly movement, several smaller modifications were introduced to the original Firefly Algorithm with the goal of improving the balance between exploration and exploitation, and ultimately reducing runtime. In the experiments, all of these modifications were applied simultaneously. As a result, it is not possible to determine which individual changes contributed most to the performance improvements—or whether any may have had a negative effect on solution quality.

To evaluate the contribution of each modification, the benchmark experiments would need to be repeated with algorithm variants in which only one modification (or subsets of them, depending on the desired level of granularity) is applied. The results could then be compared against two baselines: one with all modifications active and one using the unmodified original algorithm.

Hyperparameter Selection

One drawback of the various modifications introduced to improve the Firefly Algorithm is the resulting increase in the number of hyperparameters that must be defined before running the algorithm. In the conducted experiments, these parameters were chosen heuristically based on prior experience. No systematic procedure was employed for hyperparameter selection.

Since these parameters significantly influence the balance between exploration and exploitation, a more principled approach could improve both robustness and efficiency. This is particularly relevant in the context of MPC, where parameter estimation may need to be performed periodically. In such cases, it would be beneficial to identify hyperparameter settings that offer a good trade-off between solution quality and computational cost.

For example, in the thermal parameter estimation task for a residential building, it may be sufficient to use 50 fireflies to obtain a good temperature fit. However, if significantly more fireflies are used—simply because that setting performed well in earlier experiments—runtime may increase unnecessarily without yielding a meaningful improvement in solution quality.

One possible solution would be to use a simple grid search algorithm—or another global optimization method, depending on the dimensionality of the hyperparameter space—to identify a robust configuration prior to deployment. This tuning process could be conducted once for a specific problem setting, and the resulting hyperparameters could then be fixed for future use. This approach is particularly appropriate in cases where the structure of the optimization problem remains constant over time and only the input data changes, as is typical in MPC applications.

Applicability in an MPC Setting

Both experiments revealed that the algorithm requires a non-negligible amount of time to arrive at a solution, with runtime depending on the available computational resources. For the WWTP problem, the computation time took approximately two hours (without parallelization), while the parameter estimation for the residential building—covering 11 rooms—took around 11 hours, even with parallel processing. The latter, in particular, is a considerable time investment.

Whether the algorithm is practical for use within an MPC framework depends primarily on two factors.

The first is how frequently the parameters need to be re-estimated. If this interval is shorter than the time required for parameter estimation, the approach may not be feasible. However, this is influenced by the generalization behavior of the estimated parameters. In the WWTP case study, the parameters showed good alignment with unseen data across a two-week period, suggesting that frequent re-estimation may not be necessary. In contrast, the building case study produced more ambiguous results: some rooms exhibited overfitting, others showed poor fits, while a few generalized well. As discussed in Section 7.6.2, these inconsistencies may be due more to limitations in the experimental setup than to the algorithm itself. An improved experimental design would be needed to make stronger conclusions about re-estimation frequency.

The second factor is the computational load of the control strategy calculation itself. If computing the control input occupies most of the available time between updates, then there may be insufficient time to run parameter estimation in parallel. On the other hand, if the control strategy is computed quickly and spare computational resources are available, parameter estimation could be performed during this idle time without requiring additional hardware.

Additionally, further experiments should be conducted to evaluate how much frequent parameter re-estimation actually improves the performance of the control strategy. Most MPC setups rely on simplified models of the real-world system, which introduces an inherent mismatch between simulated and measured data. However, control strategies based on such models may still achieve significant reductions in objective metrics such as energy consumption or operational cost. It therefore remains an open question to what extent control performance is affected when using suboptimal models—either due to standard parameter settings or infrequent parameter updates.

8.1 Future Work

As discussed in both the experiment-specific and general discussion sections, there are several areas that could be improved or explored further. In general, most experiments could benefit from a refined setup. For instance, the residential building parameter estimation could be repeated with more physically informed parameter bounds, and the WWTP case study could be revisited using more realistic, noise-perturbed data.

In the following, we outline several broader directions for future work that could further enhance the algorithm's performance and applicability.

Cluster Mechanism

The clustering of fireflies into separate groups has been shown to improve the convergence rate of the algorithm with respect to the number of objective function evaluations, as demonstrated in Chapter 5 through the benchmark experiments. Furthermore, Section 4.3.2 formally showed that this approach reduces the algorithm's time complexity—from $O(n^2)$ to $O(n^{\frac{3}{2}})$ in each generation—in terms of objective function evaluations, which represent the most computationally expensive component of the optimization process.

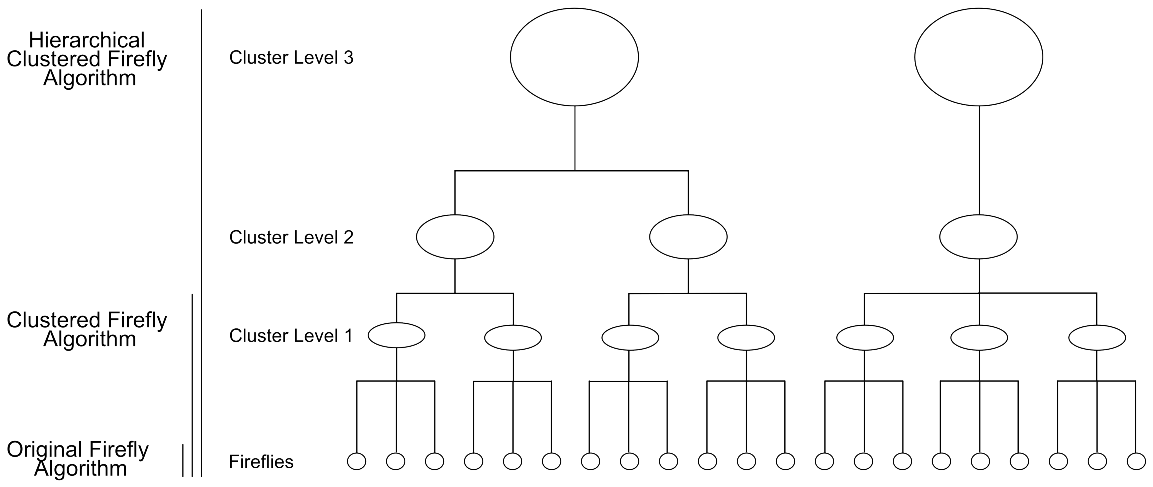

Hierarchical Clustered Firefly Algorithm
Clustered Firefly Algorithm
Original Firefly Algorithm
Cluster Level 3
Cluster Level 2
Cluster Level 1
Fireflies

Figure 8.1: Overview over the Hierarchical Clustered Firefly Algorithm idea. Different levels of clustering are displayed. At the bottom only individual fireflies are considered corresponding to the original Firefly Algorithm. At level 1, all fireflies are clustered once corresponding to the Clustered Firefly Algorithm. Additional levels could be considered for the Hierarchical Clustered Firefly Algorithm.

The current implementation employs a single layer of clustering. A natural extension would be to explore the use of multiple clustering levels. Introducing hierarchical clustering could potentially further reduce the number of required evaluations while maintaining, or even improving, estimation accuracy.

This concept is illustrated in Figure 8.1. At the base level, all fireflies are treated individually, corresponding to the original FA. In the cFA, fireflies are grouped into clusters and updated through two mechanisms: local movement within each cluster, and global movement toward the centroids of other clusters, based on average cluster intensities. This introduces a single level of clustering.

To extend this idea, additional layers of clustering can be introduced, forming a hierarchical structure. In the figure, two additional clustering levels are shown, representing a potential extension toward a hierarchical clustered Firefly Algorithm.

In such a framework, the movement rules would need to be adapted accordingly. Local movement within individual clusters would remain unchanged. At the mid-level, fireflies could consider moving toward centroids of other clusters within the same supercluster, provided those clusters have a better average intensity. At the top level, if another supercluster demonstrates superior average intensity, fireflies could additionally be influenced to move toward that supercluster's centroid. This rule-based structure can naturally be extended to accommodate deeper hierarchical levels, allowing fireflies to make movement decisions based on information aggregated at multiple scales.

Hybrid Approach

Another potential research direction, briefly discussed in Section 7.6.2, is the combination of the cFA with local optimization methods.

In such a hybrid approach, cFA could first be used for several generations to explore the global search space and identify promising regions. One possible strategy would then involve using the centroids of the final clusters as initial points for a local optimizer. This could potentially lead to faster convergence compared to running the full cFA.

An alternative strategy would be to select the best-performing m fireflies as starting points for local optimization. While these candidates are likely closer to local optima, a potential drawback is that they may be concentrated in the same region of the search space—especially if they originate from the same cluster—thus reducing the diversity of starting points and limiting the algorithm's ability to explore distinct local regions of the search space.

Both strategies offer trade-offs between exploration and exploitation and could be evaluated in future work to assess their impact on convergence speed and solution quality.

For the local optimization phase, a variety of tools can be considered, depending on the nature of the system.

Gradient-based local optimizers may be beneficial in settings where the system is smooth and well-behaved. For example, CTSM-R is a suitable choice for stochastic, continuous-time systems, particularly when dealing with noisy or uncertain measurements. However, as previously discussed, its applicability may be limited in the context of hybrid or numerically unstable systems, where the computation or approximation of derivatives becomes challenging.

In contrast, a wide range of gradient-free local optimizers are also available. These include methods such as Nelder–Mead [42], Powell's method [43], or pattern search algorithms [44], which do not rely on derivative information and are well-suited for black-box or non-smooth objective functions.
This flexibility makes the hybrid approach highly adaptable and not restricted to any single optimization strategy.

Incorporation into an MPC Framework

As stated in the introduction and the problem statement, the overarching goal of developing this parameter estimation algorithm is its application within an MPC framework for energy-intensive systems, with the aim of enabling more energy-efficient operation.

A natural next step would be to incorporate cFA into an MPC setup and study its overall impact on control performance. This could be done in collaboration with the aforementioned student project focused on optimizing WWTPs. Alternatively, cFA could be integrated into an MPC framework for residential heating systems, similar to the work presented by *Hasrat et al.* [9]. Other use cases could also be explored to further assess the practicality and effectiveness of cFA in real-world control applications.

Chapter 9

Conclusion

In this thesis, a new and effective variant of the Firefly Algorithm—*the Clustered Firefly Algorithm*—has been developed. This algorithm is a global optimization method designed to efficiently balance global exploration and local exploitation by organizing fireflies into clusters and dividing their movement into local and global movement.

It was formally shown that the algorithm achieves an improved time complexity compared to the original Firefly Algorithm, reducing the potential number of movements per generation from $O(n^2)$ to $O(n^{3/2})$, under the assumption that each cluster contains approximately $\sqrt{n}$ fireflies on average.

Experimental results on a range of benchmark optimization functions demonstrated that cFA generally converges more quickly and yields better solutions than the original Firefly Algorithm, a modified variant, and a Grid Search baseline—particularly in higher-dimensional settings.

Additionally, the algorithm was successfully applied to estimate parameters for a synthetic dataset based on a theoretical WWTP using the ASM1 model. The estimated parameters demonstrated a good model fit, even on unseen data. However, certain experimental limitations were identified, and a more realistic dataset would be necessary to fully assess the algorithm's practical applicability.

Furthermore, the algorithm was successfully applied to estimate thermal parameters of an interconnected room temperature model for a residential building, using noisy real-life data. It was compared to the parameter estimation tool CTSM-R, with cFA demonstrating greater robustness by finding parameter fits for all rooms considered. However, in some cases, the resulting fits were poor—likely due to the many assumptions, approximations, and simplifications required to implement the interconnected room model with the available data. In such cases, the model may not have been capable of accurately describing the room temperature dynamics, which reflects limitations of the implementation of the model rather than of the algorithm itself.

Lastly, in this use case, the algorithm exhibited some generalization issues, which, as discussed earlier, may stem from the experimental setup—particularly the use of unrealistic parameter bounds. To fully evaluate the effectiveness of cFA in the residential building context, further experiments with improved modeling assumptions and design might be necessary.

In relation to the problem statement and defined subgoals, the first two subgoals—concerning time complexity and optimization performance on benchmark functions—have been successfully achieved. The latter two subgoals, involving parameter estimation for the WWTP and residential building use cases, were only partially fulfilled due to the experimental limitations discussed earlier.

As a result, the overall problem statement is only partially answered. While the cFA demonstrates clear potential as a parameter estimation tool in control-oriented contexts, further experiments are necessary to fully evaluate its applicability within an MPC framework.

Bibliography

[1] IPCC. *Climate Change 2023: Synthesis Report. Summary for Policymakers*. https://www.ipcc.ch/report/ar6/syr/. Contribution of Working Groups I, II and III to the Sixth Assessment Report of the Intergovernmental Panel on Climate Change, edited by H. Lee and J. Romero. 2023. DOI: 10.59327/IPCC/AR6-9789291691647.001 (Cited on page 1).

[2] United Nations Framework Convention on Climate Change (UNFCCC). *Paris Agreement*. https://unfccc.int/sites/default/files/english_paris_agreement.pdf. Adopted at COP21 in Paris, December 12, 2015. See Article 2 for temperature targets. 2015 (Cited on page 1).

[3] Hannah Ritchie. "Sector by sector: where do global greenhouse gas emissions come from?" In: *Our World in Data* (2020). https://ourworldindata.org/ghg-emissions-by-sector (Cited on pages 1, 5).

[4] International Energy Agency. *Net Zero by 2050: A Roadmap for the Global Energy Sector*. https://www.iea.org/reports/net-zero-by-2050. Revised version, October 2021. 2021. URL: https://www.iea.org/reports/net-zero-by-2050 (Cited on page 2).

[5] James B. Rawlings, David Q. Mayne, and Moritz M. Diehl. *Model Predictive Control: Theory, Computation, and Design*. 2nd. Madison, WI: Nob Hill Publishing, 2017. ISBN: 978-0-9759377-3-0 (Cited on pages 2, 3).

[6] Yanfei Li, Zheng O'Neill, Liang Zhang, Jianli Chen, Piljae Im, and Jason DeGraw. "Grey-box modeling and application for building energy simulations - A critical review". In: *Renewable and Sustainable Energy Reviews* 146 (2021), p. 111174. ISSN: 1364-0321. DOI: https://doi.org/10.1016/j.rser.2021.111174 (Cited on page 2).

[7] Alexandre David, Peter Gjøl Jensen, Kim Guldstrand Larsen, Marius Mikučionis, and Jakob Haahr Taankvist. "Uppaal Stratego". In: *Tools and Algorithms for the Construction and Analysis of Systems*. Ed. by Christel Baier and Cesare Tinelli. Springer Berlin Heidelberg, 2015, pp. 206–211. ISBN: 978-3-662-46681-0 (Cited on pages 3, 10, 42).

[8] Manfred Jaeger, Peter Gjøl Jensen, Kim Guldstrand Larsen, Axel Legay, Sean Sedwards, and Jakob Haahr Taankvist. "Teaching Stratego to Play Ball: Optimal Synthesis for Continuous Space MDPs". In: *Automated Technology for Verification and Analysis*. Ed. by Yu-Fang Chen, Chih-Hong Cheng, and Javier Esparza. Cham: Springer International Publishing, 2019, pp. 81–97. ISBN: 978-3-030-31784-3 (Cited on page 3).

[9] Imran Riaz Hasrat, Peter Gjøl Jensen, Kim Guldstrand Larsen, and Jiří Srba. "A toolchain for domestic heat-pump control using Uppaal Stratego". In: *Science of Computer Programming* 230 (2023), p. 102987. ISSN: 0167-6423. DOI: `https://doi.org/10.1016/j.scico.2023.102987` (Cited on pages 3, 10, 50, 51, 55, 56, 60, 61, 68, 76).

[10] Xin-She Yang. "Firefly Algorithms for Multimodal Optimization". In: *Stochastic Algorithms: Foundations and Applications*. Ed. by Osamu Watanabe and Thomas Zeugmann. Springer Berlin Heidelberg, 2009, pp. 169–178. ISBN: 978-3-642-04944-6 (Cited on pages 4, 8, 12, 14, 17).

[11] Jun Li, Xiaoyu Wei, Bo Li, and Zhigao Zeng. "A survey on firefly algorithms". In: *Neurocomputing* 500 (2022), pp. 662–678. ISSN: 0925-2312. DOI: `https://doi.org/10.1016/j.neucom.2022.05.100` (Cited on page 4).

[12] International Energy Agency (IEA). *World Energy Outlook 2016*. Licence: CC BY 4.0. Paris: IEA, 2016. URL: `https://www.iea.org/reports/world-energy-outlook-2016` (Cited on pages 5, 34).

[13] M. Henze, W. Gujer, T. Mino, and M. van Loosedrecht. *Activated Sludge Models ASM1, ASM2, ASM2d and ASM3*. IWA Publishing, Oct. 2006. ISBN: 9781780402369. DOI: `10.2166/9781780402369` (Cited on pages 5, 36, 37, 42).

[14] Jorge Nocedal and Stephen J. Wright. *Numerical Optimization*. 2nd. Springer Series in Operations Research and Financial Engineering. New York: Springer, 2006. ISBN: 978-0387303031 (Cited on page 6).

[15] William H. Press, Saul A. Teukolsky, William T. Vetterling, and Brian P. Flannery. *Numerical Recipes: The Art of Scientific Computing*. 3rd. New York: Cambridge University Press, 2007. ISBN: 9780521880688 (Cited on page 6).

[16] A. Gelman, J.B. Carlin, H.S. Stern, D.B. Dunson, A. Vehtari, and D.B. Rubin. *Bayesian Data Analysis, Third Edition*. Chapman & Hall/CRC Texts in Statistical Science. Taylor & Francis, 2013. ISBN: 9781439840955 (Cited on page 7).

[17] David M. Blei, Alp Kucukelbir, and Jon D. McAuliffe. "Variational Inference: A Review for Statisticians". In: *Journal of the American Statistical Association* 112.518 (2017), pp. 859–877. DOI: `10.1080/01621459.2017.1285773` (Cited on page 7).

[18] T. Hastie, R. Tibshirani, and J.H. Friedman. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer series in statistics. Springer, 2009. ISBN: 9780387848846 (Cited on page 7).

[19] Kurt Hornik, Maxwell Stinchcombe, and Halbert White. “Multilayer feedforward networks are universal approximators”. In: *Neural Networks* 2.5 (1989), pp. 359–366. ISSN: 0893-6080. DOI: `https://doi.org/10.1016/0893-6080(89)90020-8` (Cited on page 7).

[20] M. Raissi, P. Perdikaris, and G.E. Karniadakis. “Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations”. In: *Journal of Computational Physics* 378 (2019), pp. 686–707. ISSN: 0021-9991. DOI: `https://doi.org/10.1016/j.jcp.2018.10.045` (Cited on page 7).

[21] Melanie Mitchell. *An Introduction to Genetic Algorithms*. Cambridge, MA, USA: MIT Press, 1998. ISBN: 0262631857 (Cited on page 8).

[22] J. Kennedy and R. Eberhart. “Particle swarm optimization”. In: *Proceedings of ICNN’95 - International Conference on Neural Networks*. Vol. 4. 1995, 1942–1948 vol.4. DOI: `10.1109/ICNN.1995.488968` (Cited on page 8).

[23] Malini Deepak and Rabee Rustum. “Review of Latest Advances in Nature-Inspired Algorithms for Optimization of Activated Sludge Processes”. In: *Processes* 11.1 (2023). ISSN: 2227-9717. DOI: `10.3390/pr11010077` (Cited on page 8).

[24] N A Selamat, N. A. Wahab, and S Sahlan. “Particle Swarm Optimization for multivariable PID controller tuning”. In: *2013 IEEE 9th International Colloquium on Signal Processing and its Applications*. 2013, pp. 170–175. DOI: `10.1109/CSPA.2013.6530036` (Cited on page 9).

[25] Huong Pei Choo, Shafishuhaza Sahlan, Rickey Ting Pek Eek, and Norhaliza Abdul Wahab. “Self-tuning PID controller for activated sludge system”. In: *2013 IEEE 8th Conference on Industrial Electronics and Applications (ICIEA)*. 2013, pp. 16–21. DOI: `10.1109/ICIEA.2013.6566333` (Cited on page 9).

[26] Martin Schlüter, Jose A. Egea, Luis T. Antelo, Antonio A. Alonso, and Julio R. Banga. “An Extended Ant Colony Optimization Algorithm for Integrated Process and Control System Design”. In: *Industrial & Engineering Chemistry Research* 48.14 (2009), pp. 6723–6738. DOI: `10.1021/ie8016785` (Cited on page 9).

[27] Intissar Khoja, Taoufik Ladhari, Faouzi M’sahli, and Anis Sakly. “Cuckoo Search Approach for Parameter Identification of an Activated Sludge Process”. In: *Computational Intelligence and Neuroscience* 2018.1 (2018), p. 3476851. DOI: `https://doi.org/10.1155/2018/3476851` (Cited on page 9).

[28] Xianjun Du, Junlu Wang, Veeriah Jegatheesan, and Guohua Shi. “Parameter estimation of activated sludge process based on an improved cuckoo search algorithm”. In: *Bioresource Technology* 249 (2018), pp. 447–456. ISSN: 0960-8524. DOI: `https://doi.org/10.1016/j.biortech.2017.10.023` (Cited on pages 9, 42).

[29] Seyed mohammad Ebrahimi Saryazdi, Alireza Etemad, Ali Shafaat, and Ammar M. Bahman. "Data-driven performance analysis of a residential building applying artificial neural network (ANN) and multi-objective genetic algorithm (GA)". In: *Building and Environment* 225 (2022), p. 109633. ISSN: 0360-1323. DOI: `https://doi.org/10.1016/j.buildenv.2022.109633` (Cited on page 9).

[30] Ana K Rivera, Josue Sánchez, and Miguel Chen Austin. "Parameter identification approach to represent building thermal dynamics reducing tuning time of control system gains: A case study in a tropical climate". In: *Frontiers in Built Environment* 8 (2022), p. 949426. DOI: `https://doi.org/10.3389/fbuil.2022.949426` (Cited on page 10).

[31] Rune Juhl, Jan Kloppenborg Møller, and Henrik Madsen. *ctsmr - Continuous Time Stochastic Modeling in R*. 2016. arXiv: `1606.00242 [stat.CO]`. URL: `https://arxiv.org/abs/1606.00242` (Cited on pages 10, 50, 60).

[32] J. MacQueen. "Some Methods for Classification and Analysis of Multivariate Observations". In: *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability*. Vol. 1. University of California Press, 1967, pp. 281–297 (Cited on page 19).

[33] S. Surjanovic and D. Bingham. *Virtual Library of Simulation Experiments: Test Functions and Datasets*. Retrieved April 21, 2025, from `http://www.sfu.ca/~ssurjano` (Cited on pages 22, 27).

[34] Eamonn Keogh and Abdullah Mueen. "Curse of Dimensionality". In: *Encyclopedia of Machine Learning and Data Mining*. Ed. by Claude Sammut and Geoffrey I. Webb. Boston, MA: Springer US, 2017, pp. 314–315. ISBN: 978-1-4899-7687-1. DOI: `10.1007/978-1-4899-7687-1_192` (Cited on page 28).

[35] Christoffer Brejnholm Koch, Hannes Gebauer, Lise Bech Gehlert, and Malthe Peter Højen Jørgensen. *Enhancing Effluent Quality and Energy Efficiency of the Modified Ludzack-Ettinger Process: A Model Predictive Control Approach*. Semester Project, Aalborg University. Supervised by Kim Guldstrand Larsen. Jan. 2025 (Cited on pages 35, 40, 42).

[36] Willi Gujer and Mogens Henze. "Activated Sludge Modelling and Simulation". In: *Water Science and Technology* 23.4-6 (Feb. 1991), pp. 1011–1023. ISSN: 0273-1223. DOI: `10.2166/wst.1991.0553` (Cited on pages 39, b).

[37] Imre Takács, G. G. Patry, and D. Nolasco. "A dynamic model of the clarification-thickening process". In: *Water Research* 25.10 (1991), pp. 1263–1271. DOI: `10.1016/0043-1354(91)90066-Y` (Cited on page 40).

[38] Robert Paz. "The Design of the PID Controller". In: (Jan. 2001) (Cited on page 41).

[39] Gerd Behrmann, Alexandre David, and Kim G. Larsen. *A Tutorial on Uppaal*. `https://homes.cs.aau.dk/~adavid/RTSS05/UPPAAL-tutorial.pdf`. 25/10-05 (Cited on page 42).

[40] Fritz Kasten and Gerhard Czeplak. "Solar and terrestrial radiation dependent on the amount and type of cloud". In: *Solar Energy* 24.2 (1980), pp. 177–189. DOI: 10.1016/0038-092X(80)90391-6 (Cited on page 53).

[41] Patrick Schober, Christa Boer, and Lothar A. Schwarte. "Correlation Coefficients: Appropriate Use and Interpretation". In: *Anesthesia & Analgesia* 126.5 (May 2018), pp. 1763–1768. DOI: 10.1213/ANE.0000000000002864 (Cited on page 57).

[42] John A. Nelder and R. Mead. "A Simplex Method for Function Minimization." In: *Comput. J.* 7.4 (1965), pp. 308–313. DOI: http://dx.doi.org/10.1093/comjnl/7.4.308 (Cited on page 75).

[43] M. J. D. Powell. "An efficient method for finding the minimum of a function of several variables without calculating derivatives". In: *The Computer Journal* 7.2 (Jan. 1964), pp. 155–162. ISSN: 0010-4620. DOI: 10.1093/comjnl/7.2.155 (Cited on page 75).

[44] Virginia Torczon. "On the Convergence of Pattern Search Algorithms". In: *SIAM Journal on Optimization* 7.1 (1997), pp. 1–25. DOI: 10.1137/S1052623493250780 (Cited on page 75).

Appendix A

Gujer Matrix

The Gujer matrix, shown in Table A.1, provides a compact overview of the ASM1 model. It captures the set of biological processes considered in ASM1 and how each state variable is affected by them. Each row corresponds to a biological process, while each column represents a component (state variable) or a process rate. If a component i is affected by a process j, the corresponding entry contains a stoichiometric coefficient (weight factor) indicating the extent of that influence.

The net rate of change for each component Y_i is computed as a weighted sum of the process rates:

$$\frac{dY_i}{dt} = \sum_j w_{i,j} \cdot \rho_j$$

PROCESSES		COMPONENTS													PROCESS RATES ρ_j
$j \downarrow$	$i \rightarrow$	1	2	3	4	5	6	7	8	9	10	11	12	12	
	Name	S_I	S_S	S_O	S_{NO}	S_{ND}	S_{NH}	S_{ALK}	X_I	X_{BH}	X_{BA}	X_P	X_S	X_{ND}	
1	Aerobic growth of heterotrophs		$\frac{-1}{Y_H}$	$\frac{-(1-Y_H)}{Y_H}$			$-i_{XB}$	$\frac{-i_{XB}}{14}$		1					$\mu_H \cdot \frac{S_S}{K_S+S_S} \cdot \frac{S_O}{K_{OH}+S_O} \cdot X_{BH}$
2	Anoxic growth of heterotrophs		$\frac{-1}{Y_H}$		$\frac{-(1-Y_H)}{2.86Y_H}$		$-i_{XB}$	$\frac{1-Y_H}{14 \cdot 2.86Y_H} - \frac{i_{XB}}{14}$		1					$\mu_H \cdot \frac{S_S}{K_S+S_S} \cdot \frac{K_{OH}}{K_{OH}+S_O} \cdot \frac{S_{NO}}{K_{NO}+S_{NO}} \cdot n_g \cdot X_{BH}$
3	Aerobic growth of autotrophs			$\frac{-(4.57-Y_A)}{Y_A}$	$\frac{1}{Y_A}$		$-i_{XB} - \frac{1}{Y_A}$	$-\frac{i_{XB}}{14} - \frac{1}{7 \cdot Y_A}$			1				$\mu_A \cdot \frac{S_{NH}}{K_{NH}+S_{NH}} \cdot \frac{S_O}{K_{OA}+S_O} \cdot X_{BA}$
4	Decay of heterotrophs									-1		f_P	$1-f_P$	$i_{XB} - f_P \cdot i_{XP}$	$b_H \cdot X_{BH}$
5	Decay of autotrophs										-1	f_P	$1-f_P$	$i_{XB} - f_P \cdot i_{XP}$	$b_A \cdot X_{BA}$
6	Ammonification of soluble organic N					-1	1	$\frac{1}{14}$							$k_a \cdot S_{ND} \cdot X_{BH}$
7	Hydrolysis of entrapped organics		1										-1		$k_h \cdot \frac{\frac{X_S}{X_{BH}}}{K_X + \frac{X_S}{X_{BH}}} \cdot \left(\frac{S_O}{K_{OH}+S_O} + n_h \cdot \frac{K_{OH}}{K_{OH}+S_O} \cdot \frac{S_{NO}}{K_{NO}+S_{NO}} \right) \cdot X_{BH}$
8	Hydrolysis of entrapped organic N					1								-1	$\rho_7(X_{ND}/X_S)$

Table A.1: Gujer Matrix for the ASM1 model. [36]

Appendix B

Thermal Parameter Estimation Values from Heat Pump Data Experiments

The following tables present the thermal parameters estimated in the heat pump data experiment. Table B.1 reports values estimated using cFA with the shape-weighted MSE objective. Table B.2 contains estimates obtained using cFA with a standard MSE objective. Finally, Table B.3 shows the parameters estimated using CTSM-R.

In the case of CTSM-R, entries marked with "–" indicate that no valid parameter estimates were obtained for that room. For all tables, "–" is also used to indicate parameters that do not apply to a specific room.

Table B.1: Thermal parameters estimated for each room using cFA with the shape-weighted MSE objective function

Room	T_e0	T_h0	a_a	a_e	a_h	a_s	a_w	b_h	b_h_e	a_i_0	a_i_1	a_i_2	a_i_3	a_i_4	a_i_5	a_i_6	a_i_7	a_i_12	a_i_15	a_i_17	k	set
0	12.49	22.05	2.64	469.63	28.30	849.32	60.84	499.12	480.41	–	967.63	748.69	–	28.42	4.59	7.71	–	–	25.19	2.02	8.90	19.36
1	21.99	26.76	3.40	68.74	236.87	606.22	887.10	347.18	402.29	957.52	–	610.08	83.45	2.33	3.68	504.98	420.70	11.32	21.51	6.73	6.73	20.63
2	29.04	21.34	0.56	479.55	334.64	203.37	509.57	441.95	731.75	987.58	968.83	231.80	–	81.80	272.45	98.62	554.39	153.34	93.05	74.57	7.46	20.02
3	4.10	31.44	61.92	522.60	7.11	761.82	489.38	593.59	804.57	–	730.73	–	–	–	–	–	–	–	292.11	–	4.97	20.54
4	11.56	31.77	232.77	930.82	2.55	713.26	505.63	665.53	685.72	350.59	2.96	180.73	–	–	–	485.43	488.43	484.39	957.26	–	5.59	20.60
5	17.15	25.60	14.09	804.43	23.50	665.11	230.96	59.26	627.78	324.37	13.27	370.86	–	–	–	–	–	–	–	980.32	9.74	19.09
6	19.64	24.43	8.78	87.52	125.81	744.96	412.15	67.77	818.86	430.98	850.07	28.60	–	745.93	–	–	809.41	119.91	269.74	14.54	8.12	19.05
7	24.75	35.20	5.73	397.90	521.56	184.51	168.01	544.18	447.43	–	778.57	672.35	–	57.30	–	798.18	–	772.43	58.55	115.12	6.67	19.20
12	22.26	21.65	14.66	612.79	372.02	184.21	191.46	78.88	313.92	–	44.43	231.80	–	692.51	–	116.19	936.64	–	598.82	789.83	8.84	19.31
15	28.64	25.56	352.87	645.53	0.99	258.73	450.37	725.38	567.37	251.64	442.30	159.49	680.62	992.40	–	831.38	883.27	437.74	–	279.51	4.87	21.64
17	22.74	20.33	250.97	932.95	435.90	297.16	227.75	361.57	340.82	163.64	654.96	604.82	–	–	983.39	232.33	714.51	498.35	434.63	–	9.94	19.49

Table B.2: Thermal parameters estimated for each room using cFA with the MSE objective function

Room	T_e0	T_h0	a_a	a_e	a_h	a_s	a_w	b_h	b_h_e	a_i_0	a_i_1	a_i_2	a_i_3	a_i_4	a_i_5	a_i_6	a_i_7	a_i_12	a_i_15	a_i_17	k	set
0	13.94	21.31	0.43	678.10	152.16	601.25	114.98	453.24	627.65	–	875.03	771.19	–	5.63	20.90	52.30	–	–	2.59	1.69	6.04	19.85
1	13.30	21.61	1.28	99.20	389.72	831.41	256.93	839.41	671.57	988.59	–	875.93	57.35	25.75	34.85	674.96	534.96	42.83	5.70	40.69	2.98	19.00
2	30.23	27.25	10.17	868.90	399.53	257.35	321.91	219.95	415.71	960.51	866.22	–	–	17.17	338.10	199.33	505.21	95.59	17.40	76.01	4.14	19.85
3	30.01	38.65	64.53	629.86	216.04	134.51	193.68	194.96	922.41	–	778.11	–	–	–	–	–	–	–	51.45	–	8.57	19.62
4	24.72	34.58	281.10	937.62	2.00	463.47	365.40	169.99	394.45	499.99	16.11	51.07	–	–	–	668.73	423.68	935.46	889.76	–	3.28	21.93
5	30.53	37.15	1.48	458.71	176.85	624.95	164.81	246.78	415.86	319.61	62.68	138.13	–	–	–	–	–	–	–	944.37	9.90	19.17
6	34.98	31.64	6.69	69.79	144.65	754.07	798.56	5.79	137.76	58.78	897.26	622.43	–	479.95	–	–	670.91	43.68	646.48	30.66	6.94	19.26
7	26.22	13.45	7.00	782.52	477.16	426.46	369.99	745.48	612.35	–	802.18	723.14	–	69.10	–	713.60	–	742.45	122.30	38.38	7.80	19.47
12	29.03	21.73	16.25	665.82	547.60	537.88	246.86	101.82	923.53	–	48.76	243.61	–	427.39	–	147.46	863.16	–	892.83	830.01	9.38	19.22
15	23.15	29.91	208.33	849.16	1.15	464.00	228.11	515.74	869.49	26.67	107.79	199.38	261.18	850.51	–	766.98	739.48	420.57	–	16.22	3.22	22.74
17	30.73	26.06	306.18	829.03	731.55	207.26	208.85	214.12	254.79	504.59	355.19	740.77	–	–	641.33	551.98	777.85	830.84	282.76	–	8.59	19.06

Table B.3: Thermal parameters estimated for each room using CTSM-R.

Room	T_e0	T_h0	a_a	a_e	a_h	a_s	a_w	b_h	b_h_e	a_i_0	a_i_1	a_i_2	a_i_3	a_i_4	a_i_5	a_i_6	a_i_7	a_i_12	a_i_15	a_i_17	k	set_point
0	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
1	28.515	25.994	1.4654e-05	0.089568	0.19637	0.4596	0.20031	0.21142	0.14925	9.6991	–	2.8863	0.56676	0.062165	0.46843	1.644	1.3174	0.22569	0.092077	0.13752	2.5169	21.465
2	20.347	24.246	0.00060637	0.79165	0.084519	0.003671	0.28758	0.22996	0.29924	2.0692	12.427	–	–	0.01266	2.3477	0.010711	1.3402	0.29192	0.024827	3.6071	5.0009	20.251
3	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
4	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
5	20.116	22.079	0.013175	0.18057	0.20122	0.2418	0.11631	0.12472	0.23599	0.33008	0.11682	0.47037	–	–	–	–	–	–	–	1.5313	6.3631	19.38
6	24.022	22.161	0.016038	0.15572	0.20021	0.29652	0.18608	0.18691	0.1776	0.47469	0.66775	0.35492	–	0.14914	–	–	0.351	0.19651	0.17164	0.14586	5.9117	20.058
7	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
12	16.637	18.735	0.023705	0.22857	0.21483	0.12533	0.21177	0.21241	0.2262	–	0.095491	0.13339	–	0.38276	–	0.10353	0.93438	–	0.34341	0.84842	5.4684	19.817
15	26.285	6.9125	0.16521	0.26233	0.22951	0.67105	0.028064	0.027778	0.28302	0.06151	0.14962	0.085871	0.31998	0.84998	–	0.76636	0.18869	0.27126	–	0.10219	5.4964	21.102
17	20.417	20.855	0.11986	0.20795	0.19944	0.19691	0.19099	0.19086	0.20422	0.22381	0.21892	0.21986	–	–	0.21584	0.20695	0.21377	0.20965	0.20188	–	4.852	20.138